TOSHIBA

UM-TS03***-E003

PROGRAMMABLE CONTROLLER
PROSEC T3

USER'S MANUAL
- FUNCTION -
(Ver 1.4)

Contents

Toshiba Corporation

Important information

Misuse of this equipment can result in property damage or human injury. Because controlled system applications vary widely, you should satisfy yourself as to the acceptability of this equipment for your intended purpose.

In no event will Toshiba Corporation be responsible or liable for either indirect or consequential damage or injury that may result from the use of this equipment.

No patent liability is assumed by Toshiba Corporation with respect to the use of information, illustrations, circuits, equipment, or examples of applications in this publication.

Toshiba Corporation reserves the right to make changes and improvement to this publication and/or related products at any time without notice. No obligation shall be incurred, except as noted in this publication.

This publication is protected by copyright and contains proprietary material. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means-electrical, mechanical, by photocopying, recording or otherwise-without obtaining prior written permission from Toshiba.

Copyright 1994 by Toshiba Corporation
Tokyo, Japan

Publication number: UM-TS03***-E003
1st edition June 1994

FOR SAFETY To use the T3 safely, read this section carefully before use.

1. Only use the T3 after first carefully reading this manual and related guides.

2. Do not use in any of the following environments, as they will cause malfunctions:-

   (1) Where the ambient temperature of the T3 (the temperature inside the panel) is 0℃ or below or 55℃ or above
   (2) Where the ambient humidity of the T3 (the humidity inside the panel) is 20% or less or 90% or more
   (3) Where condensation may form due to severe changes of temperature
   (4) Where there are vibration or violent shocks
   (5) Where there are corrosive gases or flammable gases
   (6) Where there is dust, salinity or iron content
   (7) Where there is direct sunlight

3. Pay attention to the following at the T3 installation site:-

   (1) For safety in maintenance and operation, keep a distance of at least 200mm from high-voltage equipment (high-voltage lines) and power equipment (power lines), or separate by a shield such as a steel plate.
   (2) Keep the expansion cables separate from other power sources when wiring. In particular, separate by at least 200 mm from high-power lines.
   (3) Provide an air space of at least 70mm around the units for ventilation.
   (4) Install the units vertically.

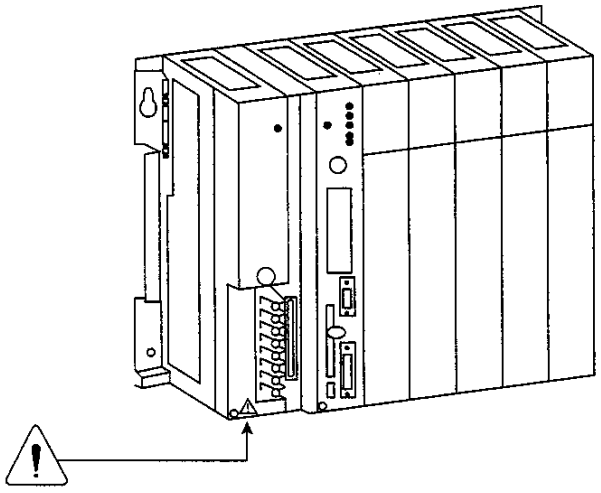
4. The T3 power supply module is a dedicated module for the T3. Do not use it for other purposes.

5. For the wiring to the module, use crimp-style terminals fitted with reverse power sheaths. When it is not possible to use crimp-style terminals fitted with sheaths, cover with insulating tape and ensure that the conducting parts are not exposed.

This is the warning mark for dangerous locations. It is attached to the equipment in positions where there is a risk of electric shock and in positions where there is a risk of damage to the equipment through wrong wiring.

Take the following precautions where there is a mark:

(1) Hazardous voltage can shock or cause severe injury if you touch the power input terminals while power on. Do not touch the power input terminals.
(2) For safety, always switch off power when wiring and during maintenance and inspections.
(3) Wire the power input terminals correctly and do not apply voltages in excess of the specified voltage limits, since this will cause the equipment damage.



Purpose of this manual

This manual describes the functions (those functions which can be achieved by the CPU and the basic hardware) of the Programmable Controller T3. This manual also provides the necessary information for designing application programs and operating the T3.
Read this manual carefully to use the T3 with it's maximum performance.

Inside of this manual

This manual is divided into the following 3 Parts.

Part 1. Basic Programming.....Gives the basic information for programming, and shows how to write a program into the T3 with a simple example.

Part 2. Functions .....For the full understanding of the T3 functions, first explains the internal operation of the T3 CPU, and then explains the detailed functions of the T3.

Part 3. Programming Information .....Explains the information for designing a program which will fully use the functions of the T3. Also explains Ladder diagram and SFC as programming languages for the T3. Explains in the detailed information summarized in Part 1.

Those who are using the T3 for the first time should first read Part 1 in order to understand the basics of programming.
When Parts 2 and 3 are read in addition, the advanced control functions of the T3 will be understood without difficulty.

Those experienced in using the T3 may skip Part 1, but refer to Parts 2 and 3 as necessary so as to fully use performance. An index is provided at the end of this manual for that purpose.

When it comes to the configuration, some of the contents of Parts 1 and 3 are duplicated. However, please note that some portions of the explanation in Part 1 are summarized for ease of understanding.

Related manuals

The following related manuals are available for the T3.

T3 User's Manual-Hardware
This manual covers the T3's main body and basic I/O-thier specifications, handling, maintenance and services.

T3 User's Manual-Functions
This document explains the functions of the T3 and how to use them. The necessary information to create user programs is covered in this volume.

T-series Instruction Set
This manual provides the detailed specifications of instructions for Toshiba's T-series Programmable Controllers.

T-PDS Basic Operation Manual
This manual explains how to install the T-series program development system (T-PDS) into your personal computer and provides basic programming operations.

T-PDS Command Reference Manual
This manual explains all the commands of the T-series program development system (T-PDS) in detail.

Handy Programmer (HP911) Operation Manual
This manual explains how to operate the Handy Programmer (HP911) for the T-series Programmable Controllers.

T-series Computer Link Function
This manual explains the specification and handling method of the T-series Programmable Controller's Computer Link function.

Analog Input Module (AD368) User's Manual
This manual provides the specifications and the operations of the Analog Input module (AD368) for the T3.

Analog Output Module (DA364/DA374) User's Manual
This manual provides the specifications and the operations of the Analog Output module (DA364/DA374) for the T3.

Note and caution symbols

Users of this manual should pay special attention to information preceded by the following symbols.

Calls the reader's attention to information considered important for full understandings of programming procedures and/or operation of the equipment.

Calls the reader's attention to conditions or practices that could damage the equipment or render it temporarily inoperative.

Terminology

AWG	American Wire Gage
ASCII	American Standard Code for Information Interchange
CPU	Central Processing Unit
EEPROM	Electrically Erasable Programmable Read Only Memory
IF	Interface
I/O	Input/Output
LED	Light-Emitting Diode
ms	millisecond
NEMA	National Electrical Manufacture's Association
PLC	Programmable Controller
PS	Power supply
RAM	Random Access Memory
ROM	Read Only Memory
μ s	microsecond
Vac	ac voltage
Vdc	dc voltage

Contents

PART 1
BASIC PROGRAMMING

1 . Overview 13
1.1 System design procedures 13
1.2 Basic programming procedures 14

2. Operation Outline 17
2.1 Operation mode and functions 17
2.2 Modes transition conditions 18
2.3 Operation flow chart 20

3. I/O Allocation 23
3.1 I/O allocation 23
3.2 Input and output registers 24
3.3 Rules for I/O allocation 26
3.4 Unit base address setting functions 29

4. User Program 31
4.1 User program configuration 31
4.2 System information 32
4.3 User program 33
4.4 Program execution sequence 35

5. User Data 36
5.1 User data types and functions 36
5.2 Conditions for data initialization 39

6. Programming Example 40
6.1 Sample system 40
6.2 Input/output allocation 41
6.3 Sample program 43
6.4 Programming procedure 47

PART 2
FUNCTIONS

1. Overview 71
1.1 T3 system configuration 71
1.2 Functional specifications 72

2. Internal Operation 73
2.1 Basic internal operation flow 73
2.2 System initialization 74
2.3 Mode control 76
2.4 Scan control 81
2.4.1 Scan mode 83
2.4.2 Batch I/O processing 85
2.4.3 Timer update 87
2.5 Peripheral support 88
2.6 Programming support functions 89

3. User Program Execution Control 92
3.1 Program types 92
3.2 Main/Sub programs execution control 93
3.3 Interrupt programs execution control 100

4. Peripheral Memory Support Functions 102
4.1 EEPROM support 102
4.2 IC memory card support 103

5. RAS Functions 104
5.1 Overview 104
5.2 Self diagnostics 104
5.3 Event history 108
5.4 Memory protect function 110
5.5 Power interruption detection function 111
5.5.1 Power interruption shut down function 111
5.5.2 Hot restart function 113
5.6 I/O error mapping function 114
5.7 Online I/O replacement function 115
5.8 Execution status monitoring 116

5.9 Sampling trace function 117
5.10 Status latch function 122
5.11 Debug support function 123
5.11.1 Force function 123
5.11.2 Online program changing function 123
5.11.3 Debug mode functions 124
5.12 System diagnostics 131
5.13 Password function 135

PART 3
PROGRAMMING INFORMATION

1. Overview 139
1.1 Aims of part 3 139
1.2 User memory configuration 139

2. User Program Configuration 141
2.1 Overview 141
2.2 System information 143
2.3 User program 147
2.3.1 Main program 148
2.3.2 Sub-program 148
2.3.3 Interrupt program 150
2.3.4 Sub-routines 153
2.4 Comments 155

3. User Data 156
3.1 Overview 156
3.2 Registers and devices 159
3.3 Register data types 182
3.4 Index modification 189
3.5 Digit designation 193

4. I/O Allocation 198
4.1 Overview 198
4.2 Methods of input/output allocation 199
4.3 Register and module correspondence 203
4.4 Network assignment 206

5. Programming Language 211
5.1 Overview 211
5.2 Ladder diagram 214
5.3 SFC 221
5.4 Programming precautions 236
5.5 List of instructions 238

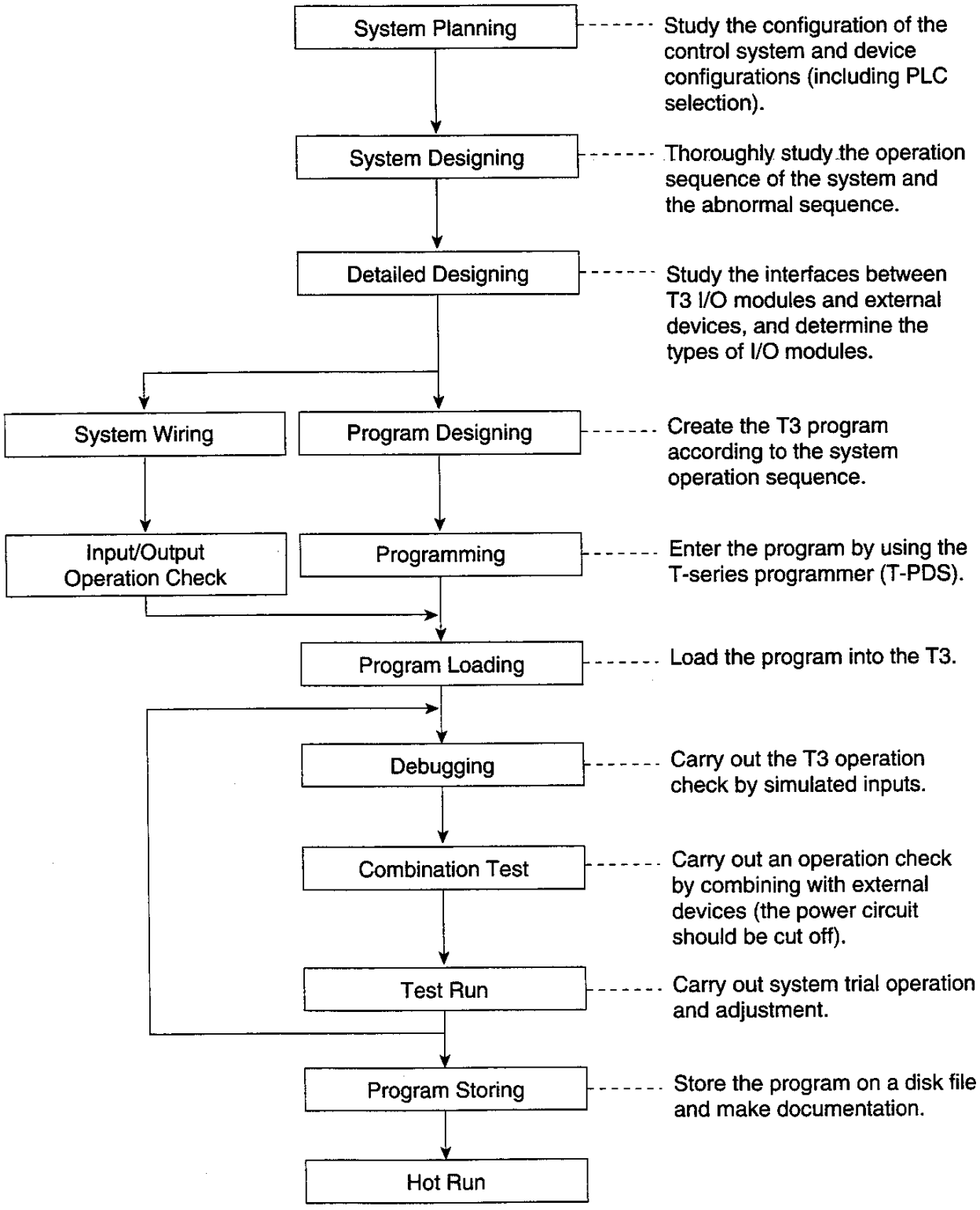
INDEX 264

PART 1

BASIC PROGRAMMING

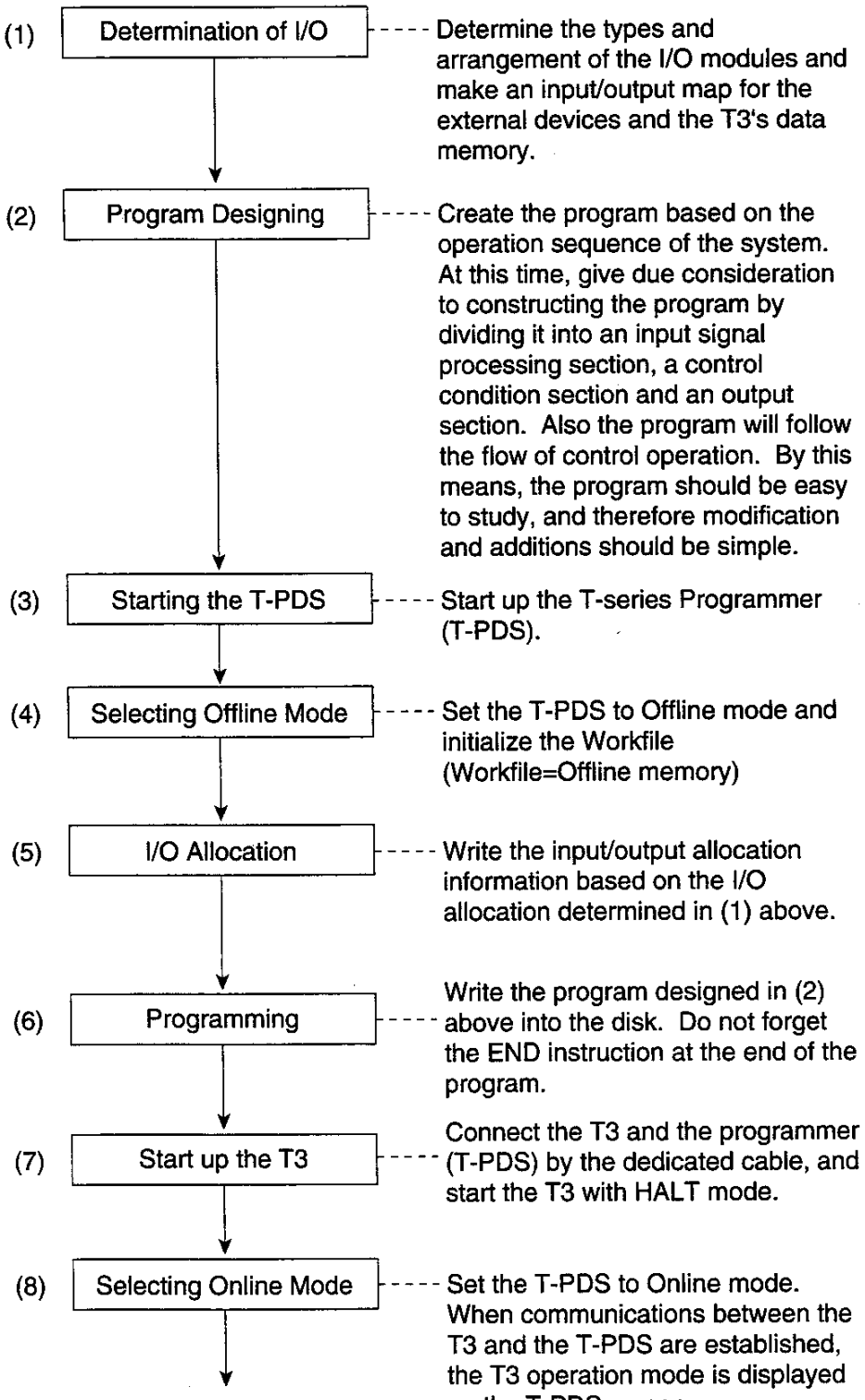
1.1 System design procedures

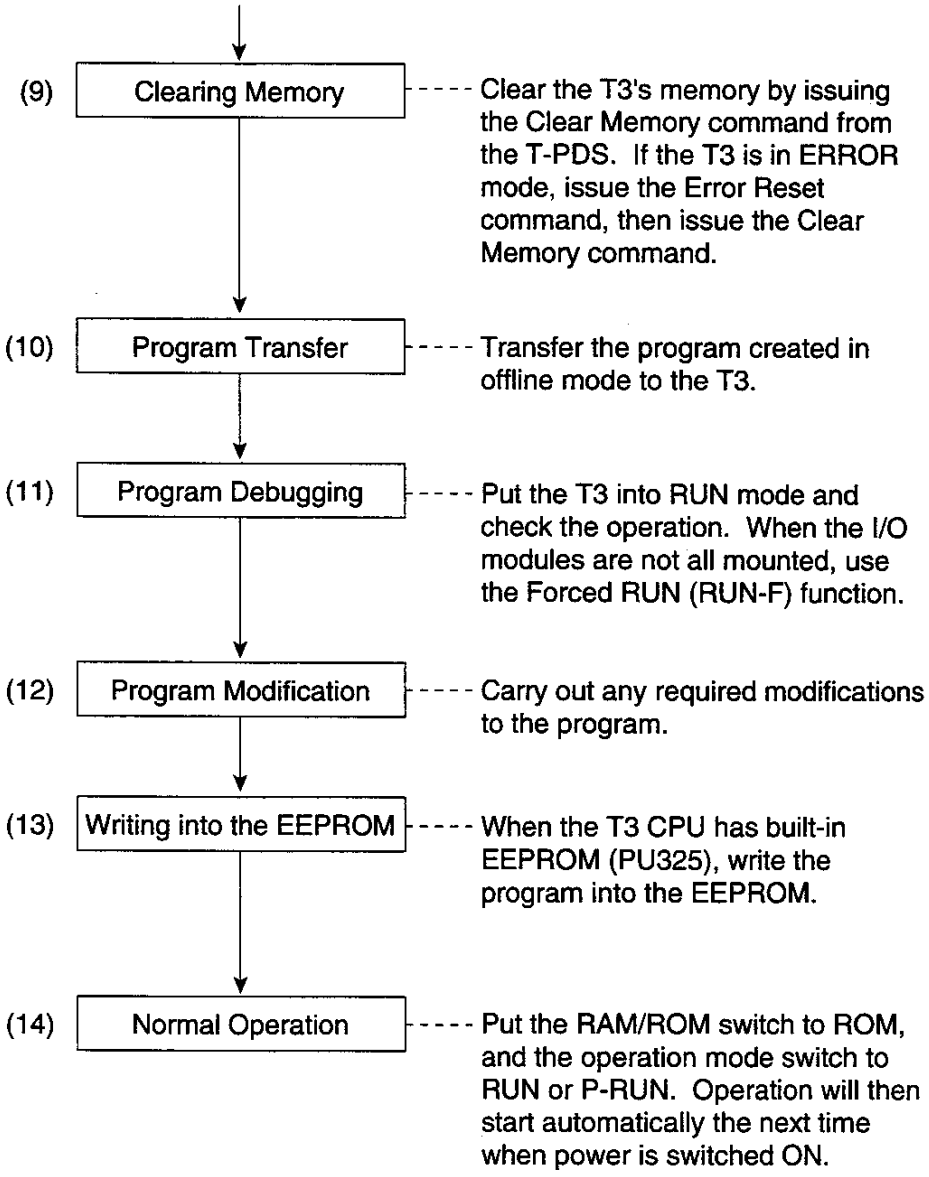
Normally, the design of a control system to which the T3 is applied is carried out by the following procedure.


System Planning
Study the configuration of the control system and device configurations (including PLC selection).
System Designing
Thoroughly study the operation sequence of the system and the abnormal sequence.
Detailed Designing
Study the interfaces between T3 I/O modules and external devices, and determine the types of I/O modules.
System Wiring
Program Designing
Create the T3 program according to the system operation sequence.
Input/Output Operation Check
Programming
Enter the program by using the T-series programmer (T-PDS).
Program Loading
Load the program into the T3.
Debugging
Carry out the T3 operation check by simulated inputs.
Combination Test
Carry out an operation check by combining with external devices (the power circuit should be cut off).
Test Run
Carry out system trial operation and adjustment.
Program Storing
Store the program on a disk file and make documentation.
Hot Run

1.2 Basic programming procedures

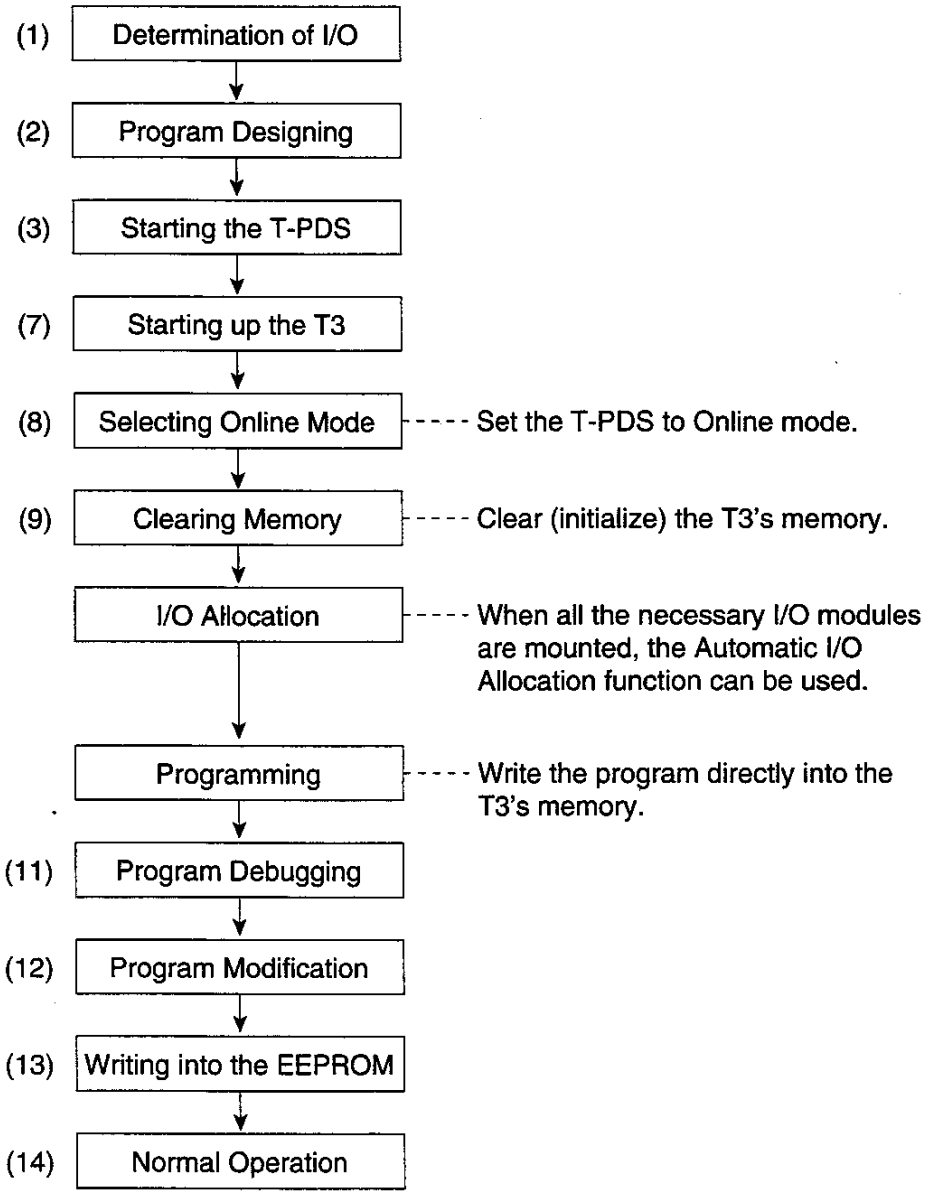
The basic procedures for creating a T3 program and loading the program into the T3 are as follows.


(1)
Determination of I/O
Determine the types and arrangement of the I/O modules and make an input/output map for the external devices and the T3's data memory.
(2)
Program Designing
Create the program based on the operation sequence of the system. At this time, give due consideration to constructing the program by dividing it into an input signal processing section, a control condition section and an output section. Also the program will follow the flow of control operation. By this means, the program should be easy to study, and therefore modification and additions should be simple.
(3)
Starting the T-PDS
Start up the T-series Programmer (T-PDS).
(4)
Selecting Offline Mode
Set the T-PDS to Offline mode and initialize the Workfile (Workfile=Offline memory)
(5)
I/O Allocation
Write the input/output allocation information based on the I/O allocation determined in (1) above.
(6)
Programming
Write the program designed in (2) above into the disk. Do not forget the END instruction at the end of the program.
(7)
Start up the T3
Connect the T3 and the programmer (T-PDS) by the dedicated cable, and start the T3 with HALT mode.
(8)
Selecting Online Mode
Set the T-PDS to Online mode. When communications between the T3 and the T-PDS are established, the T3 operation mode is displayed on the T-PDS screen.


(9)
Clearing Memory
Clear the T3's memory by issuing the Clear Memory command from the T-PDS. If the T3 is in ERROR mode, issue the Error Reset command, then issue the Clear Memory command.
(10)
Program Transfer
Transfer the program created in offline mode to the T3.
(11)
Program Debugging
Put the T3 into RUN mode and check the operation. When the I/O modules are not all mounted, use the Forced RUN (RUN-F) function.
(12)
Program Modification
Carry out any required modifications to the program.
(13)
Writing into the EEPROM
When the T3 CPU has built-in EEPROM (PU325), write the program into the EEPROM.
(14)
Normal Operation
Put the RAM/ROM switch to ROM, and the operation mode switch to RUN or P-RUN. Operation will then start automatically the next time when power is switched ON.

The above procedure is called 'Offline mode programming'.
In the Offline mode programming, after the user program is developed without the T3 hardware, it will be loaded into the T3 at a time.

On the other hand, the method of connecting the programmer (T-PDS) to the T3 and writing the program directly into the T3 is called 'Online mode programming'. The procedure of Online mode programming is as follows.


(1) Determination of I/O
(2) Program Designing
(3) Starting the T-PDS
(7) Starting up the T3
(8) Selecting Online Mode
Set the T-PDS to Online mode.
(9) Clearing Memory
Clear (initialize) the T3's memory.
I/O Allocation
When all the necessary I/O modules are mounted, the Automatic I/O Allocation function can be used.
Programming
Write the program directly into the T3's memory.
(11) Program Debugging
(12) Program Modification
(13) Writing into the EEPROM
(14) Normal Operation

NOTE

(1) Take special care for Safety during program debugging and test run.
(2) If power is switched on when the RAM/ROM switch is in RAM, the T3 will not enter RUN mode automatically even if the Operation mode switch is in RUN or P-RUN. (See Section 2.2)

2. Operation Outline

2.1 Operation modes and functions

There are 3 modes of RUN, HALT and ERROR as basic operation modes of the T3. Also, as a variation of the RUN mode, the RUN-F mode is available for debugging.

RUN Mode: This is the program execution mode. The T3 repeats the reading of external inputs, execution of the user program and the determination of external output states. (One cycle of this operation is called a 'scan'). Monitoring of the program execution state and forced input/output can be performed using the programmer.

RUN-F Mode: This is a mode to force the program execution even when the I/O modules are not mounted. (In the normal RUN mode, this would give an I/O no answer error). This is used for program debugging.

HALT Mode: This is the operation stop mode. The T3 switchs OFF all outputs and stops user program execution. Normally, programming is carried out in this mode. Also, writing the program into the EEPROM (in the case of the PU325) is available in this mode only.

ERROR Mode: This is the 'Error Down' state. When the T3 detects an error by self-diagnosis which renders continuation of operation impossible, it will switch OFF all outputs, stop the use program execution and enter the ERROR mode. In the ERROR mode, all writing operations to the T3 are prohibited. In order to escape from this mode, it is necessary either execute 'Error Reset' from the programmer, or to switch the power supply OFF and ON again.

NOTE

1. Programs can be changed in both the RUN mode and the RUN-F mode (this is called the 'online program changing function'). However, only normal programming in the HALT mode is described in Part 1. See Part 2 for the online program changing function.
2. Apart from the above 4 modes, there are actually the HOLD mode and the DEBUG mode as well. These are described in Part 2.

2.2 Modes transition conditions

To determine/change the operation mode of the T3, the operation mode switch on the CPU module, programmer PLC control commands and T3 self-diagnosis are available. Also, the RAM/ROM switch on the CPU module controls the operation mode at power up. These are described below.

* Operation Mode Switch...HALT/RUN/P-RUN

Switch Position	Operation Mode
HALT	When the mode switch is shifted from RUN or P-RUN to HALT, the operation mode will turn to the HALT mode. Also, when power is switched ON with the mode switch at HALT, the T3 will start up in the HALT mode.
RUN	When the mode switch is shifted from HALT to RUN, the operation mode will turn to the RUN mode. When the switch is shifted from P-RUN to RUN, the operation mode will not change. The mode when power is switched ON in the RUN position will be determined by the RAM/ROM switch.
P-RUN	Signifies 'Protect RUN'. Although its relationship with the operation mode is the same as that of the RUN position, in the case of P-RUN, the whole program and the leading 4k words (D0000-D4095) of the data register will be write-protected. Also, 'Initial Load' is prohibited.

* RAM/ROM Switch

Switch Position	Operation Mode
RAM	Starts up in the HALT mode regardless of the setting of the operation mode switch.
ROM	Automatic RUN will start when the operation mode switch is in RUN or P-RUN.

* Programmer PLC Control Commands...HALT/RUN/Force RUN

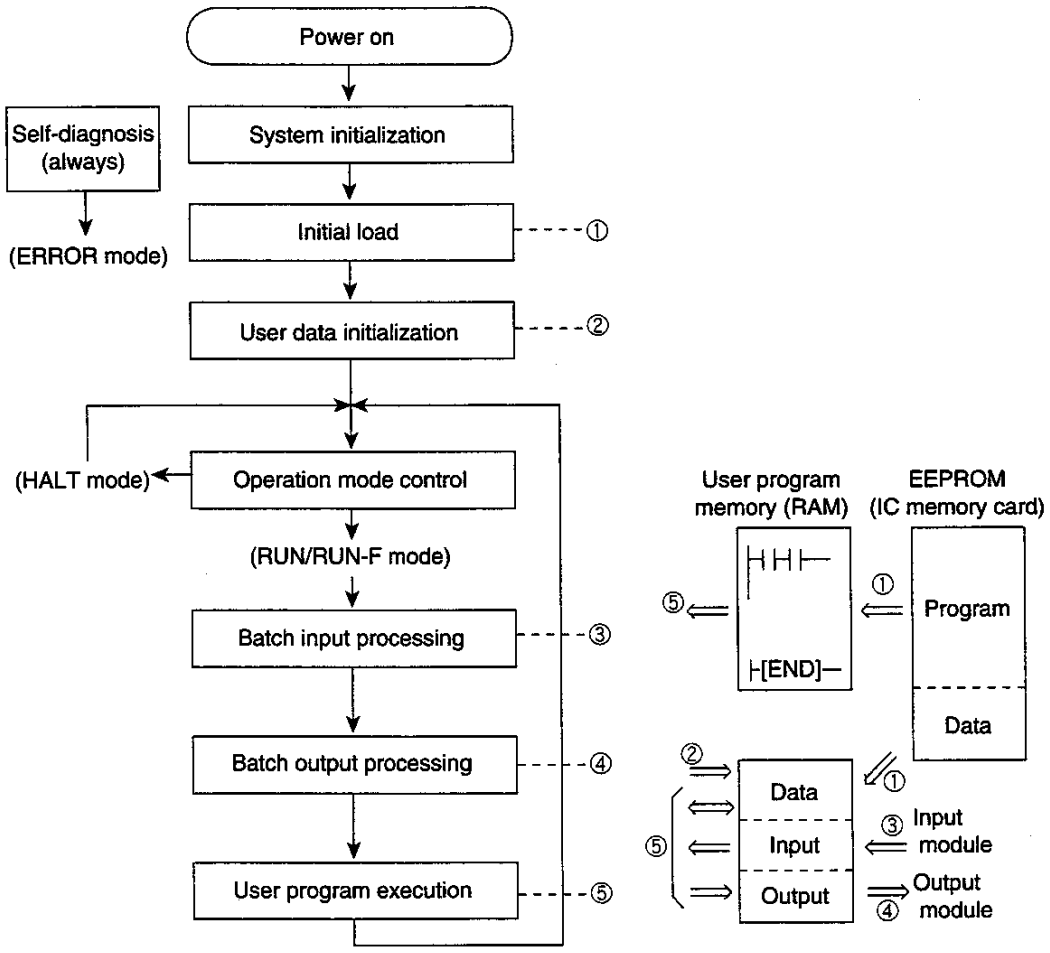
Command	Operation Mode After Execution of Command
HALT	Shifts to the HALT mode (effective only when the mode switch is in RUN or P-RUN).
RUN	Shifts to the RUN mode (effective only when the mode switch is in RUN or P-RUN).
Force RUN	Shifts to the RUN-F mode (effective only when the mode switch is in RUN or P-RUN).

Previous state			OP mode transition factor	OP mode after transition	Remarks
OP mode	RAM/ROM	Mode SW			
— (Power OFF)	RAM	HALT/RUN	Power ON	HALT	No Initial Load
		P-RUN	Power ON	HALT	No Initial Load (Protect)
	ROM	HALT	Power ON	HALT	Initial Load execution
		RUN	Power ON	RUN	Initial Load execution→RUN
		P-RUN	Power ON	RUN	No Initial Load (Protect)
	—	—	Error detection at power ON	ERROR	
HALT	RAM	HALT	Mode SW →RUN	RUN	No Initial Load
		RUN/P-RUN	Command RUN	RUN	
			Command Force RUN	RUN-F	
	ROM	HALT	Mode SW →RUN	RUN	Initial Load execution→RUN
		RUN	Command RUN	RUN	
			Command Force RUN	RUN-F	Initial Load execution→RUN-F
		P-RUN	Command RUN	RUN	No initial Load (Protect)
			Command Force RUN	RUN-F	
	—	RUN	Mode SW →HALT	HALT	Mode unchange
			Mode SW →P-RUN	HALT	Mode unchange (Protect)
		P-RUN	Mode SW →RUN	HALT	Mode unchange (Protect release)
		HALT	Command (any)	HALT	Command invalid (Mode unchange)
		RUN/P-RUN	Command HALT	HALT	
		—	Error detection	ERROR	
RUN	—	RUN	Mode SW →HALT	HALT	
			Mode SW →P-RUN	RUN	Mode unchange (Protect)
		P-RUN	Mode SW →RUN	RUN	Mode unchange (Protect release)
		RUN/P-RUN	Command HALT	HALT	
			Command RUN	RUN	Command invalid (Mode unchange)
			Command Force RUN	RUN	
			Error detection	ERROR	
RUN-F	—	RUN	Mode SW →HALT	HALT	
			Mode SW →P-RUN	RUN-F	Mode unchange (Protect)
		P-RUN	Mode SW →RUN	RUN-F	Mode unchange (Protect release)
		RUN/P-RUN	Command HALT	HALT	
			Command RUN	RUN-F	Command invalid (Mode unchange)
			Command Force RUN	RUN-F	
			Error detection	ERROR	
ERROR	—	—	Mode SW(HALT/RUN/P-RUN)	ERROR	Invalid
			Command (except Error Reset)	ERROR	
			Command Error Reset	HALT	Recovery to HALT mode

1) In this table, OP mode, RAM/ROM and Mode SW mean Operation mode, RAM/ROM switch and Operation Mode switch, respectively.
2) — means the switch status is not related to.
3) See next page for the Initial Load.

2.3 Operation flow chart

User programs can be produced without fully understanding the internal processes of the T3. However, understanding the outline of the internal processes will be effective in producing more efficient programs and in carrying out appropriate debugging. The following drawing gives a T3 internal process overview.



① Initial Load

When the RAM/ROM switch is in ROM and the operation mode switch is in other than P-RUN, the following contents stored in the EEPROM (PU325 only) or the IC memory card will be transferred to the T3 RAM at power up and at transiting from the HALT mode to the RUN mode.

In the case of an IC memory card being mounted in a EEPROM-type T3 (PU325), the initial load will not be executed from the EEPROM, the initial load will be executed from the IC memory card only.

(1) Whole user program
(2) Leading 4k words of data register (D0000 to D4095)

② User Data Initialization
User data (data register, timer, counter, input register, output register, etc) are initialized. User data is explained in Section 5.

③ Batch Input Processing
The status of external input signals will be read from input modules and stored in the input registers. (The input register is sometimes called the 'input image table'.)

④ Batch Output Processing
The status of output registers is written to the output modules. The output module determines the ON/OFF state of output based on this. (The output register is sometimes called the 'output image table'.)

⑤ User Program Execution
The instructions stored in the user program memory are read one by one, and the contents of the output register are updated while referring to the contents of the user data. This is an essential function of the T3.

One cycle from operation mode control to user program execution is called 'one scan'. The time required for 1 scan is called the 'scan cycle' (or the 'scan time').

Generally, the shorter the scan cycle, the faster the output response to a change in input signal.

NOTE

The important items related to the T3 operation mode and the switches are summarized below.

(1) When power is turned on with the RAM/ROM switch at RAM position, the T3 starts up in HALT mode. Therefore, use the RAM position during debug and test run, and set to ROM in normal operation, regardless of the type of the T3 CPU.

(2) When the Operation Mode switch is in P-RUN, leading 4k words of the data register (D0000 to D4095) will be write-protected. For some instructions, data writing to this area by program execution is also disabled. (refer to Part 2 Section 5.4)

(3) In a CPU with a built-in EEPROM (PU325) or an IC memory card, the Initial Load will be executed at power on and at the beginning of RUN mode transition if the RAM/ROM switch is in ROM and the Operation Mode switch is in other than P-RUN. Therefore, if you have changed the RAM program, write it into the EEPROM or the IC memory card before turning off power or switching to RUN mode.

(4) The object of the Initial Load is whole program and the leading 4k words of data register (D0000 to D4095).
Therefore, even if the range of D0000 to D4095 is specified as retentive, these data will be initialized by the data of the EEPROM or the IC memory card.

(5) When the Initial Load condition is satisfied, do not change the Operation Mode switch HALT → RUN → P-RUN quickly. If so, the Initial Load will be interrupted and RAM program become abnormal. Turn to P-RUN only after the RUN LED is lit.

3.1 I/O allocation

As described in Section 2.3, communication between input modules or output modules and the user program is executed via the input registers and the output registers.

I/O allocation is the determination of which address of the I/O registers shall be assigned to which I/O module. Basically, this is determined by the mounting order of the modules. Therefore, informing the CPU of the module mounting order is called 'I/O allocation'.

The following two methods are available for performing I/O allocation. Either method requires that the T3 is in the HALT mode and that the operation mode switch is in a position other than P-RUN.

(1) Automatic I/O Allocation
Execute the automatic I/O allocation command to the T3 from the programmer. The T3 CPU reads the module types of I/O modules mounted (see the table on the next page) and stores this in the user program memory as I/O allocation information.

(2) Manual I/O Allocation
Set the mounting positions and the module types of I/O modules on the I/O allocation screen of the programmer, and write this information to the T3.
Manual I/O allocation is used when performing programming in a state in which not all the I/O modules have been mounted, or when using the unit base address settings described in Section 3.4.
Manual I/O allocation is also used for offline mode programming.

When the I/O allocation information is stored in the T3 memory by these methods, the correspondence between the I/O modules and the I/O register is automatically determined by the rules described in Section 3.3.

*) In practice, special allocation of module types other than those shown in the table on the next page can be executed by manual I/O allocation. However, the description is omitted here. The details are described in Part 3.

The module type of I/O module is expressed in the following table by a combination of a functional classification (X:Input, Y:Output, X+Y:I/O mixed) and the number of registers occupied (W).

Module	Description	Module Type
DI334/334H	32 points DC input	X 2W
DI335/335H	64 points DC input	X 4W
IN354/364	32 points AC input	X 2W
DO333	16 points DC output	Y 1W
DO334	32 points DC output	Y 2W
DO335	64 points DC output	Y 4W
AC363	16 points AC output	Y 1W
AC364	32 points AC output	Y 2W
RO364	32 points Relay output	Y 2W
RO363S	16 points Relay output (isolated)	Y 1W
AD368	8 channels analog input	X 8W
DA364/374	4 channels analog output	Y 4W
CD332	Change detect 8 points DC input	iX 1W
PI312	2 channels pulse input	iX+Y 2W
AS311	ASCII module	iX+Y 4W
SN321/322/323	TOSLINE-S20 data transmission	TL-S
MS311	TOSLINE-F10 data transmission	TL-F

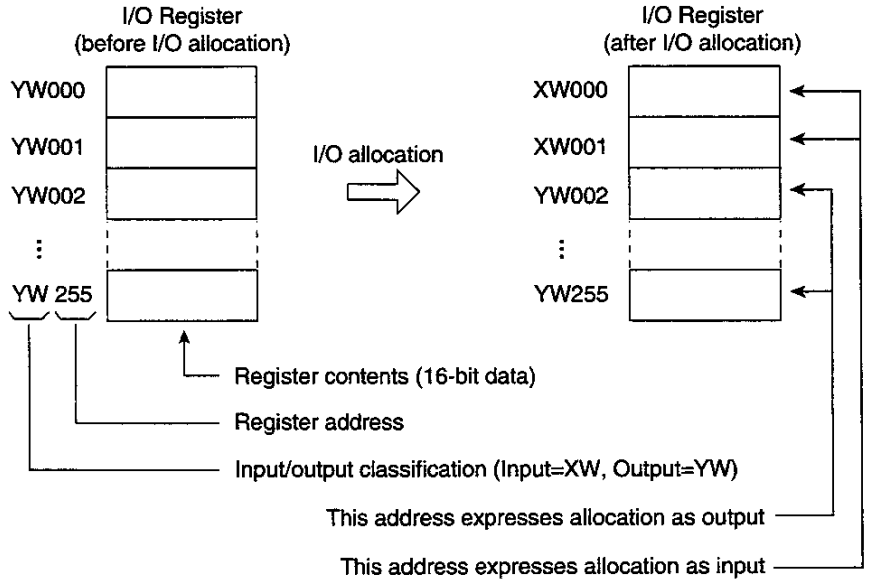
3.2 Input and output registers

In the previous Section, I/O allocation is the performance of correspondence between I/O modules and input/output registers. Here, the configurations of input registers and output registers, and methods of address expression are described.

In descriptions hitherto, input registers and output registers have been treated as separate entities. However, from the viewpoint of memory configuration, this is not correct.

In practice, the input register and the output register use the same memory area which is called the 'I/O register'. In other words, before performing I/O allocation, the I/O register is not colour-divided for input and output. Colour-division of input and output in register units (16-bit units) is performed by carrying out I/O allocation. (Before allocation, internally, all are regarded as output registers).

This idea can be conveyed by the following drawing.



The I/O register is a 16-bit register, and 256 registers are available. ('16-bit' signifies that it stores the ON/OFF information for 16 points.)

The I/O register used in the user program is expressed as follows.

When an input register ...XW□□□
When an output register ...YW□□□

The above □□□ expresses the register address (also called the 'register number'), a decimal number from 000 to 255.

Also, each bit (called a 'device') in the I/O register is expressed as follows.

When a bit in an input register (input device) ...X□□□□
When a bit in an output register (output device) ...Y□□□□

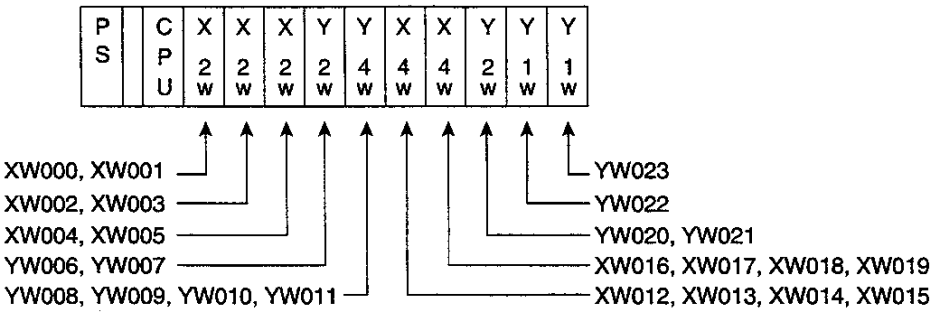
The above □□□ expresses the register address and the ▣ expresses the bit position in the register.

As bit positions, 16 positions of 0, 1, ..., 9, A, B, C, D, E, F are available.

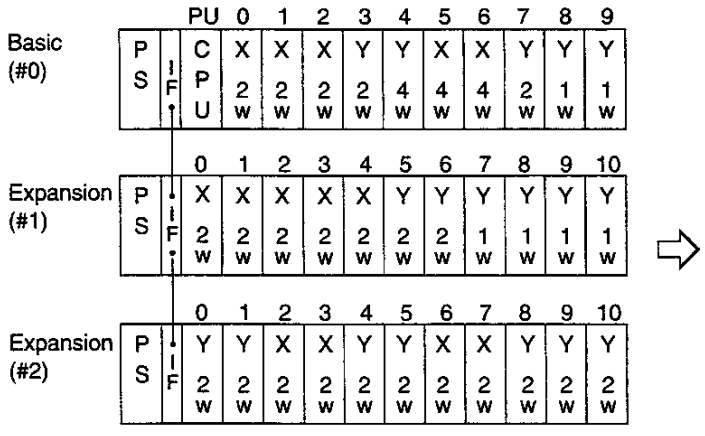
3.3 Rules for I/O allocation

When I/O allocation is performed either by the automatic I/O allocation or the manual I/O allocation method, the I/O allocation information (information on which type of module is mounted in which position) is produced in the user program memory. The coordination between the registers and the I/O modules is decided according to the following rules.

(1) In the basic unit, allocation is carried out from the module immediately to the right of the CPU in sequence from the lowest register address.



(2) In the case of expansion units, allocations are given following on from the previous stage unit in sequence from the left end module to the right end module.

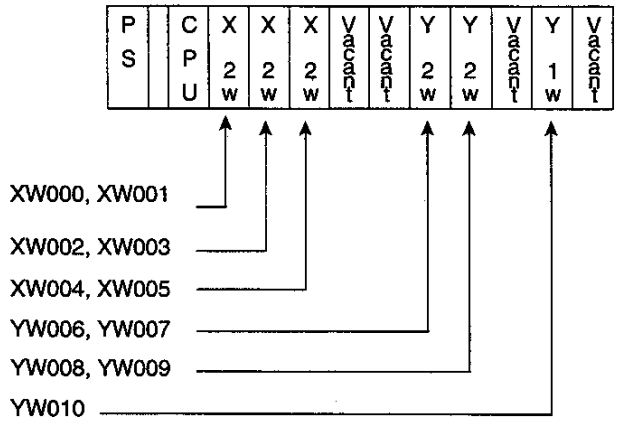

PU 0 1 2 3 4 5 6 7 8 9
Basic (#0)
PS IF CPU X2w X2w X2w Y2w Y4w X4w X4w Y2w Y1w Y1w
0 1 2 3 4 5 6 7 8 9 10
Expansion (#1)
PS IF X2w X2w X2w X2w X2w Y2w Y2w Y1w Y1w Y1w Y1w
0 1 2 3 4 5 6 7 8 9 10
Expansion (#2)
PS IF Y2w Y2w X2w X2w Y2w Y2w X2w X2w Y2w Y2w Y2w

Register allocation table

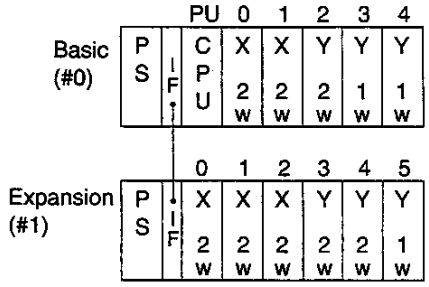
Unit	Slot	Type	Register
0	0	X2W	XW000, XW001
	↓	↓	↓
	9	Y1W	YW023
1	0	X2W	XW024, XW025
	1	X2W	XW026, XW027
	2	X2W	XW028, XW029
	3	X2W	XW030, XW031
	4	X2W	XW032, XW033
	5	Y2W	YW034, YW035
	6	Y2W	YW036, YW037
	7	Y1W	YW038
	8	Y1W	YW039
	9	Y1W	YW040
	10	Y1W	YW041
2	0	Y2W	YW042, YW043
	↓	↓	↓

*) In the I/O allocation, for convenience, the module mounting position is expressed by a combination of the unit number and the slot number.
Unit number: #0, #1, #2, #3 in sequence from the basic unit
Slot number: 0, 1, 2, ... 9 (or 10) in sequence from the module mounting position at the left end.

(3) Slots in which no module is mounted (in manual I/O allocation, slots for which no type is set) do not occupy registers. These are called 'vacant' slots,



(4) In case of the 5-slot basic rack (BU315), slots 5 to 9 are regarded as vacant. Similarly, in case of the 6-slot expansion rack (BU356), slots 6 to 10 are regarded as vacant.

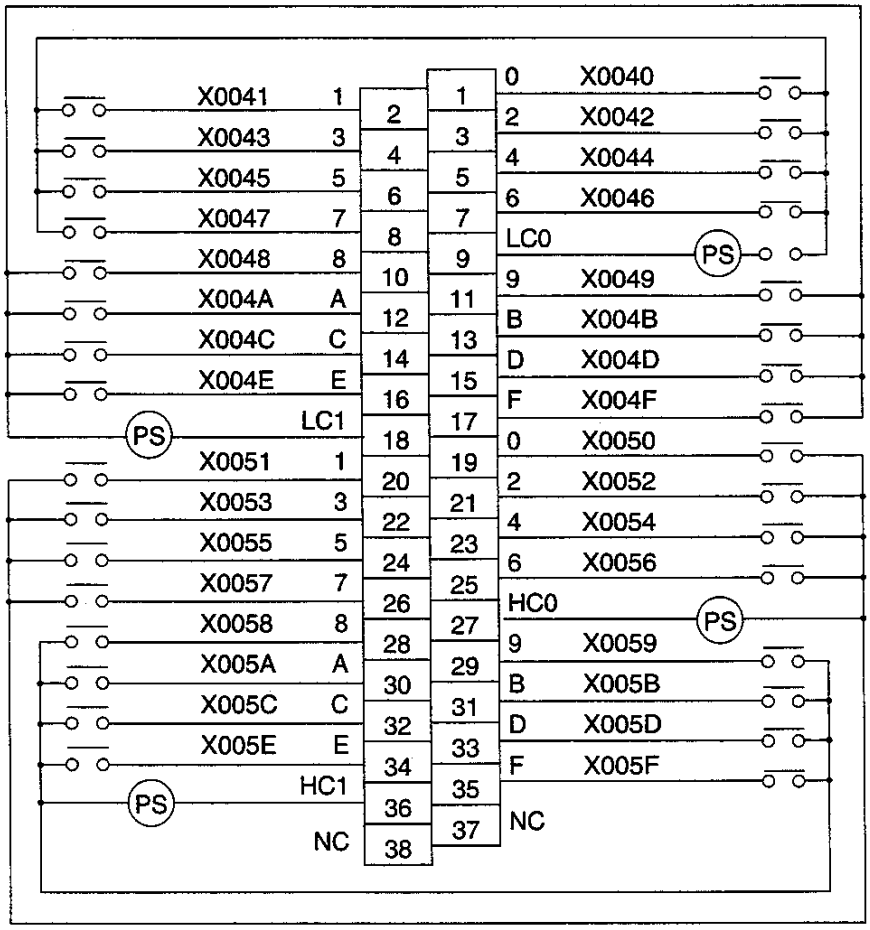


Register allocation table

Unit	Slot	Type	Register
0	0	X 2W	XW000, XW001
	1	X 1W	XW002, XW003
	2	Y 2W	YW004, YW005
	3	Y 1W	YW006
	4	Y 1W	YW007
	5	Vacant	—
	↓	↓	↓
	9	Vacant	—
1	0	X 2W	XW008, XW009
	1	X 2W	XW010, XW011
	2	X 2W	XW012, XW013
	3	Y 2W	YW014, YW015
	4	Y 2W	YW016, YW017
	5	Y 1W	YW018
	6	Vacant	—
	↓	↓	↓
	10	Vacant	—

(5) After an input/output register is allocated to an I/O module, the individual external signals on the module are allocated to each bit (device) on the register. At this time, in modules to which multiple registers are allocated, lower register address is allocated to the lower common (LC) side.

(Example)
The following is the input signal and input device coordination when XW004 and XW005 are allocated to a 32-point input module (X2W).



(6) Special modules (modules which are not designated by X, Y, X+Y, iX, iY, iX+Y as module types) such as data transmission modules do not occupy input/output registers.

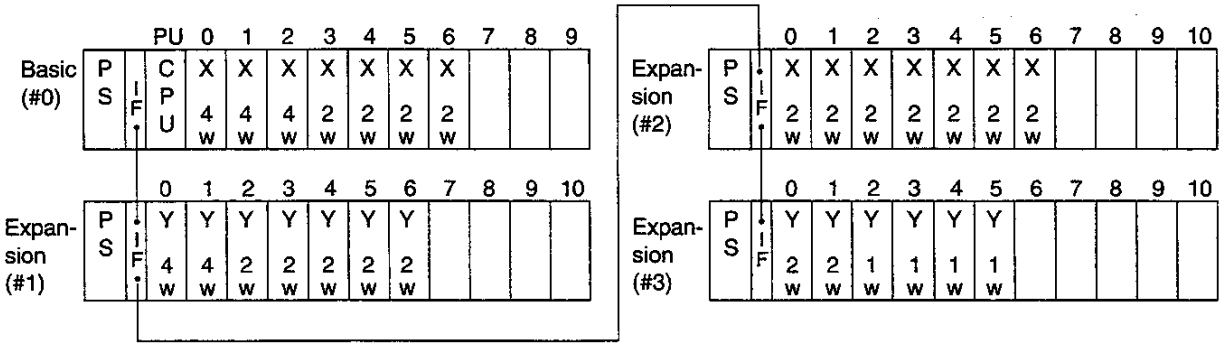
(7) Input/output registers which are not allocated, internally become output registers, and can be used in the same way as auxiliary registers/relays in the program.

3.4 Unit base address setting functions

As a special function for input/output allocation, there is a function which can set the base register address of each unit.

This function is achieved by the manual I/O allocation.

If this function is used, the register address does not shift even when module additions are carried out in the future.


Basic (#0)
PS
I/F
CPU
PU 0 1 2 3 4 5 6 7 8 9
X 4w X 4w X 4w X 2w X 2w X 2w X 2w
Expansion (#1)
0 1 2 3 4 5 6 7 8 9 10
Y 4w Y 4w Y 2w Y 2w Y 2w Y 2w Y 2w
Expansion (#2)
0 1 2 3 4 5 6 7 8 9 10
X 2w X 2w X 2w X 2w X 2w X 2w X 2w
Expansion (#3)
0 1 2 3 4 5 6 7 8 9 10
Y 2w Y 2w Y 1w Y 1w Y 1w Y 1w

Register allocation table

Unit	Unit base address	Slot	Type	Register
0	000	PU		—
		0	X 4W	XW000~XW003
		1	X 4W	XW004~XW007
		2	X 4W	XW008~XW011
		3	X 2W	XW012, XW013
		4	X 2W	XW014, XW015
		5	X 2W	XW016, XW017
		6	X 2W	XW018, XW019
		7		—
		8		—
		9		—
1	050	0	Y 4W	YW050~YW053
		1	Y 4W	YW054~YW057
		2	Y 2W	YW058, YW059
		3	Y 2W	YW060, YW061
		4	Y 2W	YW062, YW063
		5	Y 2W	YW064, YW065
		6	Y 2W	YW066, YW067
		7		—
		8		—
		9		—
		10		—

Unit	Unit base address	Slot	Type	Register
2	100	0	X 2W	XW100, XW101
		1	X 2W	XW102, XW103
		2	X 2W	XW104, XW105
		3	X 2W	XW106, XW107
		4	X 2W	XW108, XW109
		5	X 2W	XW110, XW111
		6	X 2W	XW112, XW113
		7		—
		8		—
		9		—
		10		—
3	150	0	Y 2W	YW150, YW151
		1	Y 2W	YW152, YW153
		2	Y 1W	YW154
		3	Y 1W	YW155
		4	Y 1W	YW156
		5	Y 1W	YW157
		6		—
		7		—
		8		—
		9		—
		10		—

NOTE

(1) Apart from register address skipping between units, when the unit base address setting function is used, it follows the I/O allocation rules described in Section 3.3.

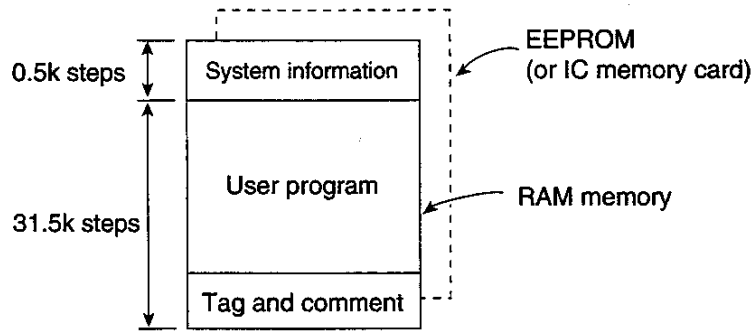
(2) A setting which gives a latter stage unit a low register address cannot be performed. For example, a setting by which the base address of Unit #1 is 50 and the base address of Unit #2 is 30 cannot be performed.

(3) When automatic I/O allocation is performed, there is no base address designation for any unit. The registers are allocated in succession. (As described in Section 3.3).

4.1 User program configuration

A group of instructions for executing control is called a 'user program'. This is also called an 'application program', a 'sequence program' or a 'logic circuit'. In this manual it will be called a 'user program'.

The memory area which stores the user program is called the 'user program memory', and in the T3 it has a capacity of 32k steps. However, out of this, 0.5k steps are used to store the user program ancillary information (this is called 'system information'). Therefore, the actual user program capacity will be 31.5k steps. Also, if Tags and Comments are stored in the T3, a part of this area is used. A 'step' is the minimum unit which composes an instruction and, depending on the type of instruction, there will be 1-10 steps per instruction.



NOTE

(1) The EEPROM is available only in the case of a built-in EEPROM type CPU (PU325). When an IC memory card is installed, the IC memory card will take priority over the EEPROM, but the description is omitted in Part 1. See Part 2 for the IC memory card.
(2) For the conditions for transfer from the EEPROM (or the IC memory card) to the RAM (the Initial Load), see Section 2.3.
(3) Tag and Comment are explained in Part 3.

4.2 System information

'System information' is the area which stores execution control parameters and user program control information for executing the user program, and occupies 0.5k steps. The following contents are included in the system information.

(1) Machine parameters (model type, memory capacity)
(2) User program information (program ID, system comments, number of steps used, etc.)
(3) Execution control parameters (scanning mode, sub-program and interrupt program execution conditions)
(4) Retentive memory area information
(5) I/O allocation information
(6) I/O interrupt assignment information
(7) Network assignment information
(8) Computer link parameters
(9) System diagnosis function execution conditions

Out of these, the CPU automatically performs the setting/updating of the machine parameters of (1) and the number of steps used of (2). Items apart from these are set by the user from the programmer. Here, only the retentive memory area information of (4) and the I/O allocation information of (5) are described. The other items are described in Part 2 and Part 3.

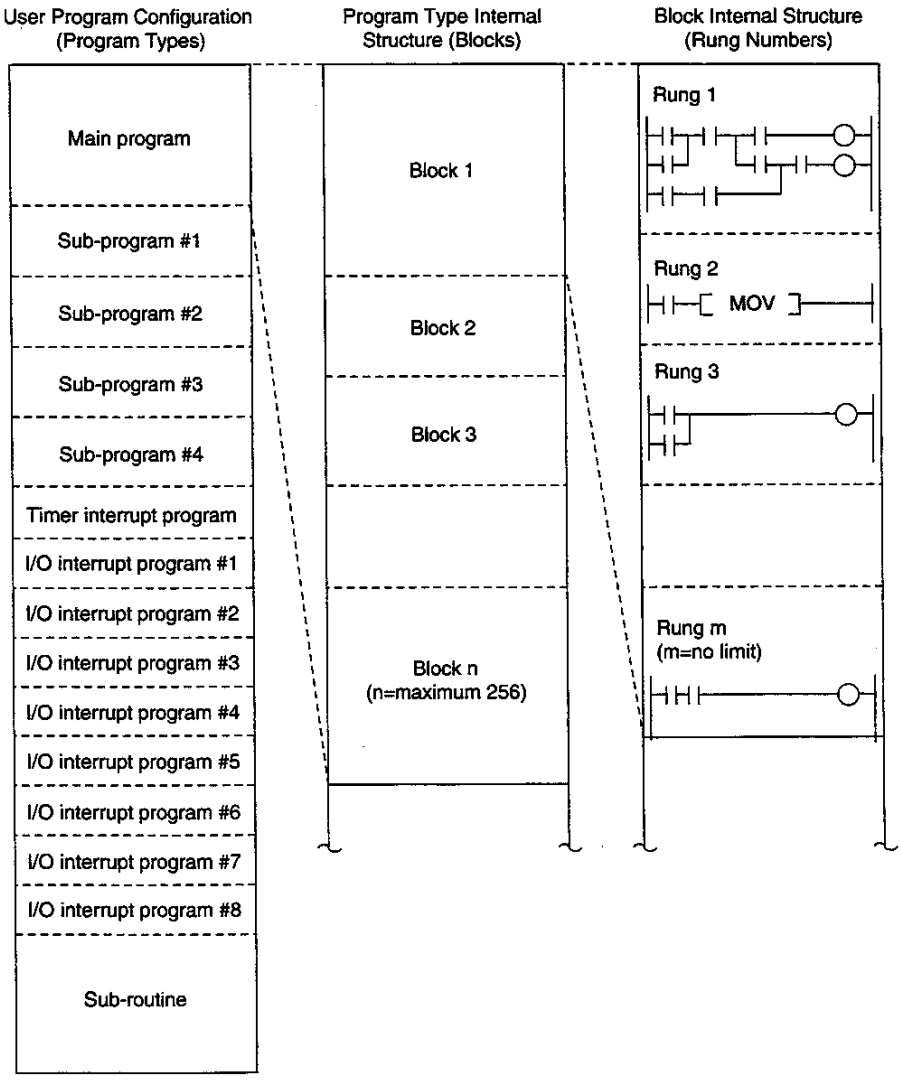
* Retentive memory area
  The ranges for retaining the data during power off can be set for the auxiliary register (RW), the timer register (T), the counter register (C) and the data register (D). Data other than within these set ranges will be 0-cleared (device is OFF) in the data initialization process at power up. This setting is performed in a way to designate from the first address (0) to a designated address for each of the above registers. (See Section 5.2 for details)

* I/O allocation information
  As described in Section 3, I/O allocation information is stored here by executing automatic I/O allocation or manual I/O allocation. The CPU determines input/output register allocation based on this information. Also, as self-diagnosis, the CPU executes a check as to whether the modules in the allocation information are correctly mounted.

4.3 User program

The user program is a group of instructions for executing control, and has a capacity of 31.5k steps. The function which executes the user program is the main function of the programmable controller T3.

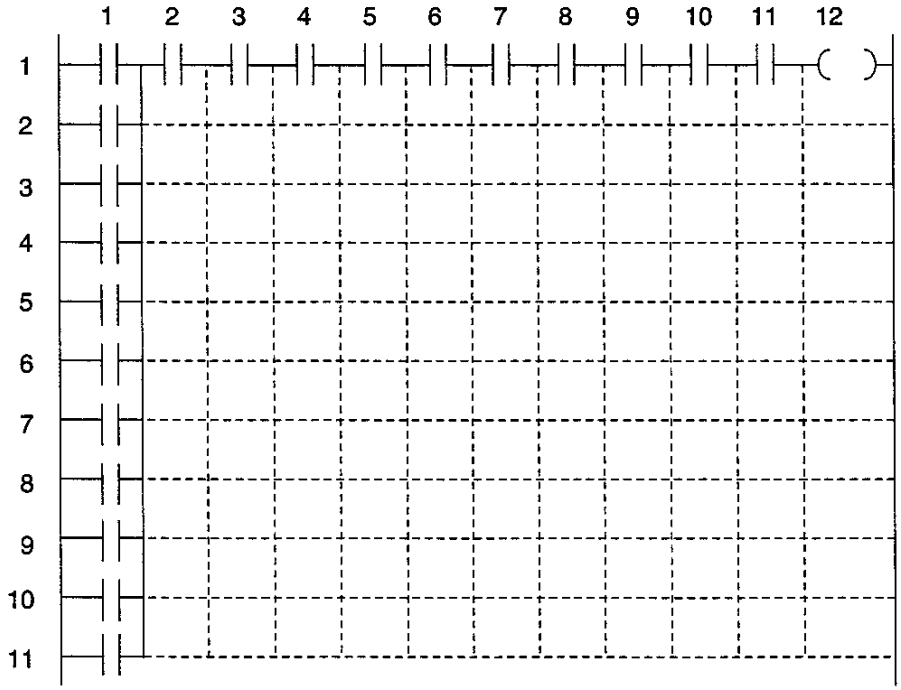
The user program is stored by each program type as shown in the following diagram, and it is managed by units called 'blocks' in each program type. Also, in 1 block, the user program is managed by a rung number (in the case of ladder diagram). Therefore, in the monitoring/editing the user program, a specified rung can be called by designating the program type, block number and rung number.



* Program Types
  As program types, the main program, sub-programs (#1-#4), the timer interrupt program, I/O interrupt programs (#1-#8) and the sub-routines are available. Although there is a capacity limit of within a total of 31.5k steps, there is no capacity limit on any of the program types.

* Blocks
  From 1 to 256 are effective as block numbers. Every block has no capacity limit. In the T3, apart from the Ladder diagram, the SFC language can be used. However multiple languages cannot be used in one block. In other words, when multiple languages are used, it is necessary to separate blocks. In the case of using the ladder diagram only, there is no need to divide the block.

* Rungs
  Within the block, the user program is managed by the rung number. (In the case of the Ladder diagram). A 'rung' signifies one grouping which is linked by lines other than right and left power rails. There is no limit to the number of rungs which can be programmed within one block. The size of one rung is limited to 11 lines x 12 rows (maximum 132 steps), as shown in the following diagram.



4.4 Program execution sequence

The main program is the main body of the user program which executes every scan, and must have at least one END instruction. Here, the program execution sequence is described in the case of the main program only. The operation of other program types is described in Part 2.

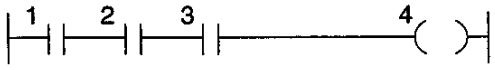
The user program is executed in the following sequence.

① The main program is executed in sequence from the first block (the lowest number block) to the block which contains the END instruction.

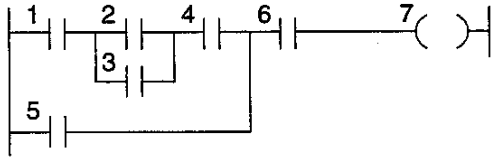
② Within one block, it is executed in sequence from the first rung (Rung 1) to the last rung (in the case of the block containing the END instruction, to the rung which has the END instruction).

③ Within one rung, it is executed in accordance with the following rules.

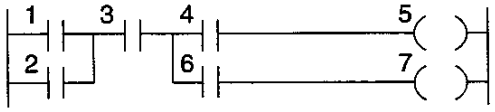
(1) When there is no vertical connection, execution is carried out from left to right.



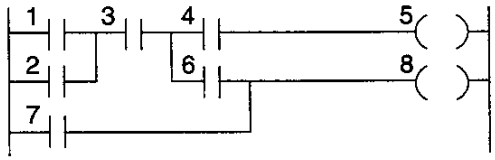
(2) When there are OR connections the OR logic path is executed first.



(3) When there are branches, execution is carried out from the upper line to the lower line.



(4) A combination of (2) and (3) above.



NOTE

1. The block numbers need not be consecutive. In other words, there may be vacant blocks in the middle.
2. The rung numbers must be consecutive. In other words, vacant rungs cannot be programmed in the middle.

5.1 User data types and functions

Data stored in the RAM memory of the CPU and which can be referred directly in a user program, such as the states of input/output signals, control parameters and arithmetical results during execution of the user program are called 'user data'.

From the viewpoint of treatment, user data can be considered as divided into registers and devices.

Registers are locations which store 16-bit data. The following types are available according to their functions.

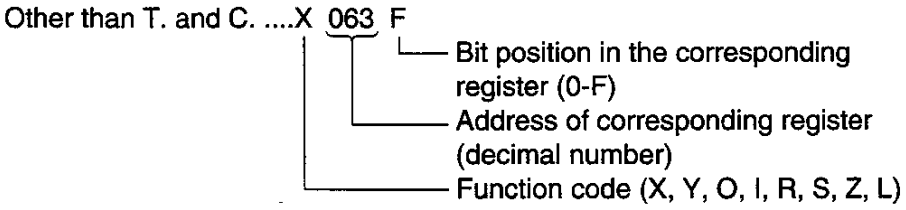
Code	Name	Function	Number	Address Range
XW	Input register	Stores input data from the input module (batch input)	Total 256 words	XW000-XW255
YW	Output register	Stores output data to the output module (batch output)		YW000-YW255
IW	Direct input register	Direct input data from the input module (direct input)		IW000-IW255
OW	Direct output register	Direct output data to the output module(direct output)		OW000-OW255
RW	Auxiliary register	Used as a temporary memory for results during execution of the user program	512 words	RW000-RW511
SW	Special register	Stores error flags, execution control flags, clock-calender data timing clocks, etc.	256 words	SW000-SW255
T	Timer register	Stores elapsed time during timer instruction execution.	512 words	T000-T511
C	Counter register	Stores current count value during counter instruction execution	512 words	C000-C511
D	Data register	Used for storing control parameters and as a temporary memory for execution results	8192 words	D0000-D8191
W	Link register	Data exchange area with data transmission module (TOSLINE-S20)	1024 words	W0000-W1023
LW	Link relay register	Data exchange area with data transmission module (TOSLINE-F10)	256 words	LW000-LW255
F	File register	Used for storing control parameters and for storing accumulated data	8192 words	F0000-F8191
I	Index register	Used for indirect addressing for register designation of instructions	1 word	I (No address)
J			1 word	J(No address)
K			1 word	K (No address)

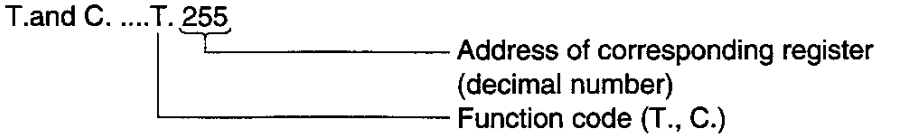
*1) In the T3 system, 1 word is treated as equal to 16 bits and units called words are used as numbers of registers.
*2) All register addresses are decimal numbers.
*3) In the timer register T000-T063 increase in 0.01 second units (0.01 second timer) and T064-T511 increase in 0.1 second units (0.1 second timer).

On the other hand, 'devices' are locations which store 1-bit data (ON/OFF information). The following types are available according to their functions.

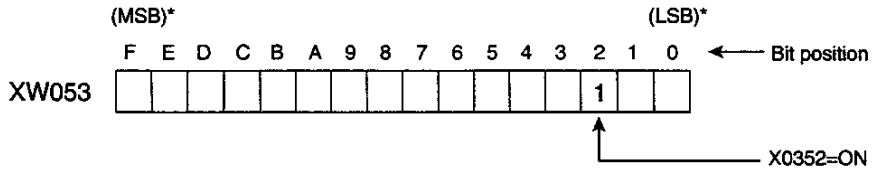
Code	Name	Function	Number	Address Range
X	Input device	Stores input data from the input module (batch input) Corresponds to 1 bit in the XW register	Total 4096 points	X0000-X255F
Y	Output device	Stores output data to the output module (batch output) Corresponds to 1 bit in the YW register		Y0000-Y255F
I	Direct input device	Direct input data from the input module (direct input)		I0000-I255F
O	Direct output device	Direct output data to the output module(direct output)		O000-O255F
R	Auxiliary relay device	Used for intemal relay. Corresponds to 1 bit in the RW register	8192 points	R000-R511F
S	Special device	Stores error flags, execution control flags, timing relays, etc. Corresponds to 1 bit in the SW register	4096 points	S0000-S255F
T.	Timer relay device	Reflects the execution result of the timer instruction Corresponds to the T register operation of the same address	512 points	T.000-T.511
C.	Counter relay device	Reflects the execution result of the counter instruction Corresponds to the C register operation of the same address	512 points	C.000-C.511
Z	Link device	Data exchange area with data transmission module (TOSLINE-S20) Corresponds to 1 bit in the leading 512 words of the W register	8192 points	Z0000-Z511F
L	Link relay device	Data exchange area with data transmission module (TOSLINE-F10)	4096 points	L0000-L255F

The address expressions for devices are as shown below.


Other than T. and C. ....X 063 F
Bit position in the corresponding register (0-F)
Address of corresponding register (decimal number)
Function code (X, Y, O, I, R, S, Z, L)



Therefore, for example, device X0352 expresses bit 2 of register XW035, and if X0352 is ON, it means that bit 2 of XW035 is 1.



NOTE

(1) The least significant bit (LSB) is bit 0 when numerical values are handled in the register.
(2) When the direct input register/device(IW/I) are used in an instruction, input data will be read directly from the input module when that instruction is executed. (This system is called the 'direct input system'). As opposed to this, in the input register (XW), input data will be read from the corresponding input module in a batch before user program execution. (This system is called the 'batch input system'). In the input/output allocation, an IW and XW of the same address correspond to the same input module.
(3) When the direct output register/device (OW/O) are used in an instruction, those data will be outputted directly to the output module when that instruction is executed. (This system is called the 'direct output system'). As opposed to this, the contents of the output register (YW) will be outputted to the corresponding output module in a batch before user program execution. (This system is called the 'batch output system').
In the input/output allocation, an OW and YW of the same address correspond to the same output module.
Note that, in the case of direct output by device O, the other 15 bits in the same register (OW) are also directly outputted.
(4) See Part 3 for details of registers/devices.

* LSB: Least significant bit
MSB: Most significant bit

5.2 Conditions for data initialization

The user data are initialized according to the conditions in the following table at power up and at transiting the RUN mode.
Also, the leading 4k words of the data register (D0000 to D4095), are the subjects of the Initial Load. Therefore, when the Initial Load conditions are established, initialization will be carried out in the sequence Initial Load → data initialization. (See Section 2.3 for Initial Load)

Register/Device	Initialization
Input registers/devices(XW/X)	For forced input devices the previous state is maintained, the others are 0-cleared
Output registers/devices(YW/Y)	For coil forced output devices the previous state is maintained, the others are 0-cleared.
Auxiliary registers/devices(RW/R)	For registers designated as retentive and coil forced devices the previous state is maintained, the others are 0-cleared
Special registers/devices(SW/S)	CPU setting part is initialized and the user setting part is maintained.
Timer registers/relays (T/T.)	For registers designated as retentive and the devices which correspond to them the previous state is maintained, the others are 0-cleared
Counter registers/relays (C/C.)	
Data registers (D)	For registers designated as retentive the previous state is maintained, the others are 0-cleared. If the Operation Mode switch is in P-RUN, leading 4k words (D0000 to D4095) are maintained.
Link registers/relays (W/Z)	For forced link devices the previous state is maintained, the others are 0-cleared
Link relays (LW/L)	For forced link relays the previous state is maintained, the others are 0-cleared
File registers (F)	All maintained
Index registers (I,J,K)	All 0-cleared

*) The retentive memory area designation is available for the RW, T, C and D registers.
These areas are designated by the system information setting function of the programmer. For each register the area from the first address (0) to the designated address becomes the retentive memory area.

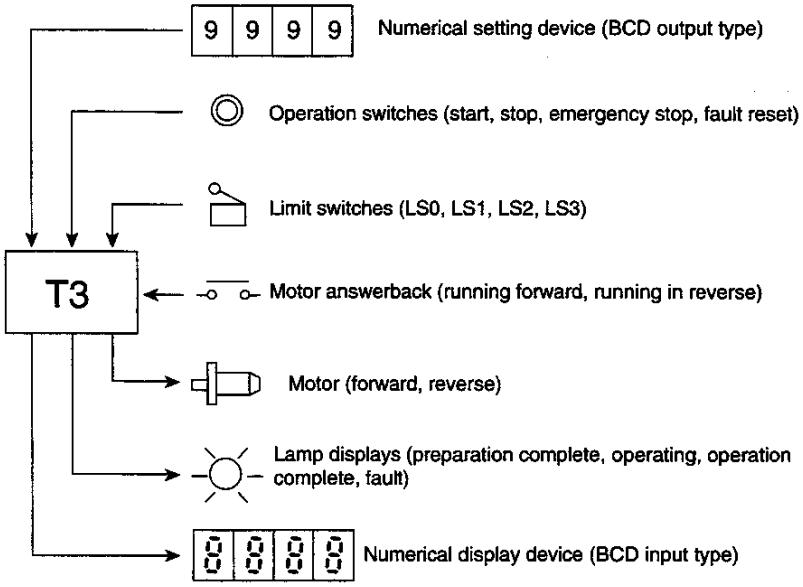
T-PDS's Retentive Memory Area Designation Screen

```
13. Retentive memory area
        RW000   ~   [    ]
        T000    ~   [    ]
        C000    ~   [    ]
        D0000   ~   [    ]
```

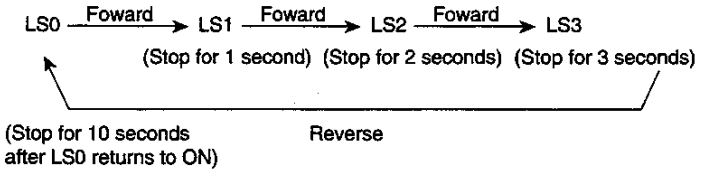
6.1 Sample system

In this section, simple sequences as examples, input/output allocation, program designing and also the procedures for the actual programming operation are shown. Refer to them when using the T3.

Let us consider the sequence in the following diagram as an example


9 9 9 9
Numerical setting device (BCD output type)
Operation switches (start, stop, emergency stop, fault reset)
Limit switches (LS0, LS1, LS2, LS3)
T3
Motor answerback (running forward, running in reverse)
Motor (forward, reverse)
Lamp displays (preparation complete, operating, operation complete, fault)
Numerical display device (BCD input type)

① When the 'Start' switch is pressed with LS0 in the ON state, the following operation is executed.



② The above operation is repeated only for the number of times set by the numerical setting device. During the operation, the 'Operating' lamp is lit and, at the same time, the actual number of executions at that time is displayed on the numerical display device.

When the operation is completed, the 'Operating' lamp will go out, and the 'Operation complete' lamp will be lit.

③ If the 'Stop' switch is pressed during the operation, the motor is stopped at that position and, after 1 second, starts in reverse. When the LS0 becomes ON, the motor is stopped and, after 1 second, the 'Preparation complete' lamp is lit.

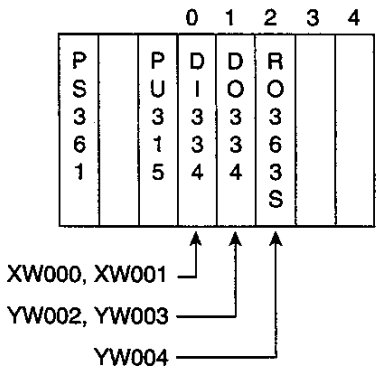
④ When LS0 is ON in states other than during operation, the 'Preparation complete' lamp is lit. The 'Start' switch is only effective when the 'Preparation complete' lamp is lit.

⑤ When the 'Emergency stop' switch has been pressed, the motor is stopped in that position and the 'Fault' lamp is lit. In that state, if the 'Fault reset' switch is pressed, the 'Fault' lamp will go out.

6.2 Input/output allocation

First decide the module configuration and make a Map of Correspondence between external signals and registers/devices. Here, the allocation is made for modules with the configuration shown below.

* Module configuration and register allocation



Rack : BU315 (for Basic, 5 I/O slots)
PS : PS361 (100-120/200-240 Vac)
CPU : PU315 (Standard)
Input : DI334 (32 points DC input)
Output : DO334 (32 points DC output)
RO363S (16 points isolated Relay output)

* Input/Output Map

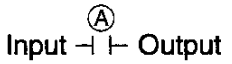
XW000 (Numerical Setting Device)		XW001 (Switches)	
X0000	$\times 10^0$	X0010	Emergency stop (normally ON)
01		11	Fault reset
02		12	Start
03		13	Stop
04	$\times 10^1$	14	
05		15	
06		16	
07		17	
08	$\times 10^2$	18	LS0
09		19	LS1
0A		1A	LS2
0B		1B	LS3
0C	$\times 10^3$	1C	Answerback forward
0D		1D	Answerback reverse
0E		1E	
0F		1F	

YW002 (Numerical Display Device)		YW003 (Lamps)		YW004 (Motor)	
Y0020	×10⁰	Y0030	Fault	Y0040	Forward
21		31	Preparation complete	41	Reverse
22		32	Operating	42	
23		33	Operation complete	43	
24	×10¹	34		44	
25		35		45	
26		36		46	
27		37		47	
28	×10²	38		48	
29		39		49	
2A		3A		4A	
2B		3B		4B	
2C	×10³	3C		4C	
2D		3D		4D	
2E		3E		4E	
2F		3F		4F	

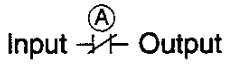
6.3 Sample program

A sample program of this sequence are shown on the following pages. When designing a program, arrange the conditions, and give them careful thought so that the program will follow the flow of operations as far as possible.

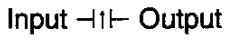
Here, the program is composed using basic instructions only. The following is a simple explanation of the instructions used in this program.



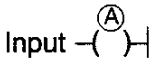
NO contact
Put output ON when the input is ON and the state of device Ⓐ is ON.



NC contact
Put output ON when the input is ON and the state of device Ⓐ is OFF.

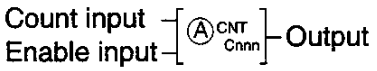


Transitional contact (rising)
Put output ON only when the input at the previous scan was OFF and the input at the present scan is ON.



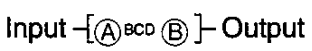
Coil
Put device Ⓐ ON when the input is ON, and put device Ⓐ OFF when the input is OFF.

ON-delay timer
After the input has changed from OFF to ON, put output ON after the elapse of the time specified by Ⓐ. Also, at this time put the corresponding timer relay ON. (the 0.1 second timer in the example on the next page)

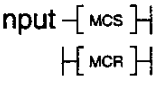


Counter
With the enable input in the ON state, count the number of times the count input is ON and store in counter register Cnnn. When the values of Ⓐ and Cnnn become equal, put output ON. When the enable input is OFF, clear Cnnn and put output OFF.

Binary conversion
When the input is ON, convert the value of BCD which has been stored in Ⓐ to a binary number and store in Ⓑ.

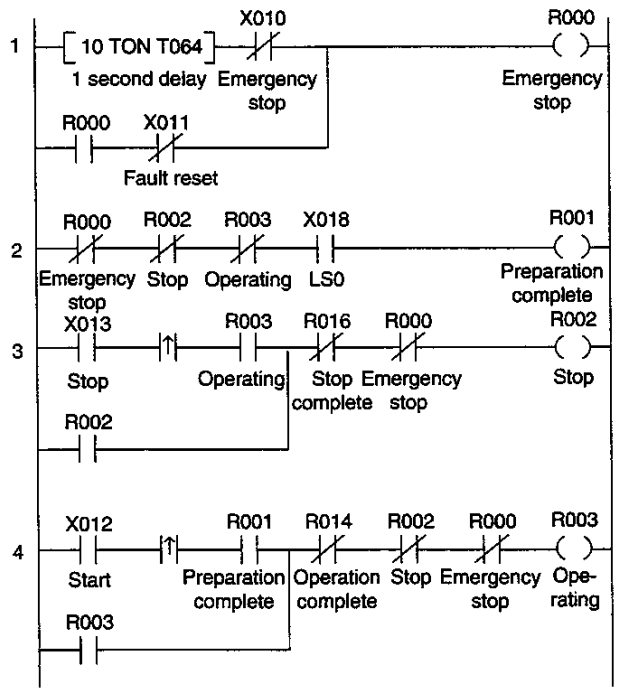


BCD conversion
When the input is ON, convert the value of Ⓐ to BCD and is store in Ⓑ.



Master control set/reset
Put the power rail between MCS and MCR ON only when the input of MCS is ON.

Operation Mode Setting Part



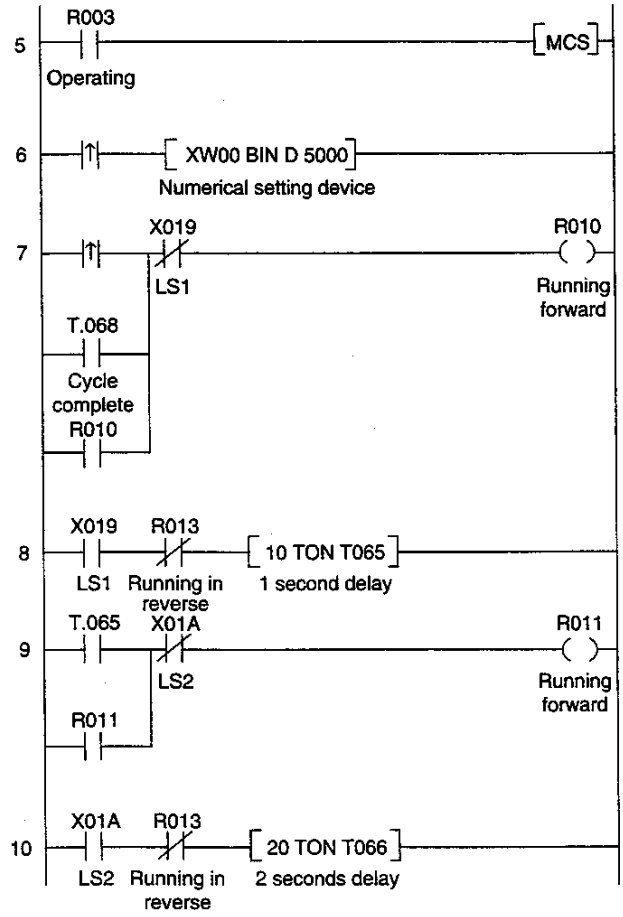
When 'Emergency stop' (X010: normally ON) is OFF, maintains R000 ON. R000 is reset by 'Fault reset' (X011) ON.

ON delay timer is to wait the establishment of the emergency stop signal.

When 'operating' (R003) is ON and 'Stop' (X013) is ON,the stop mode (R002) is ON. When 'Stop complete' (R016) is ON, R002 is put to reset.

When 'Preparation complete' (R001) is ON and 'Sart' (X012) is ON,the operation mode (R003) is put ON. When 'Operation complete'(R014) is ON, R003 is put to reset.

Operating Sequence



When the operation mode (R003) is ON, the MCS-MCR is executed.

The value of the numerical setting device is binary-converted at the beginning of the operation, and is substituted in D5000.

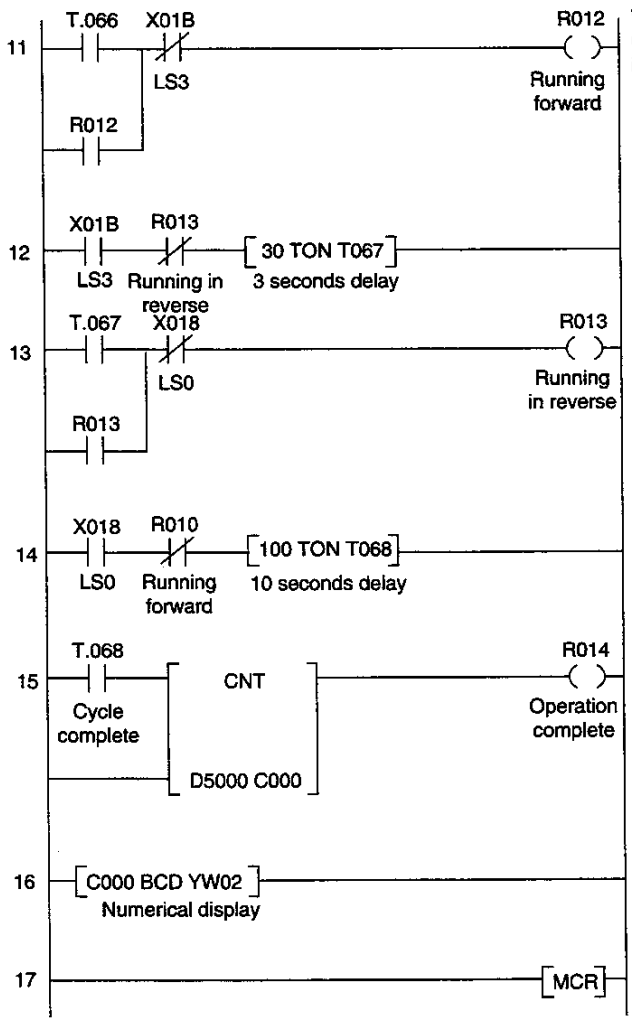
R010 is put ON at the beginning of the operation. R010 is reset by LS1 (X019) being ON. When'Cycle complete' (T.068) is ON also, in the same way, R010 is put ON.

T.065 is put ON after LS1 (X019) has been ON for 1 second.

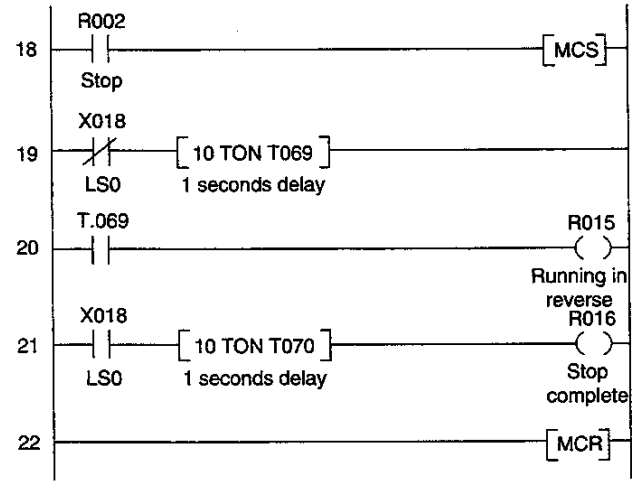
When T.065 is ON, R011 is put ON and held. R011 is reset by LS2 (X01A) being ON.

T.066 is put ON after LS2 (X01A) has been ON for 2 seconds.

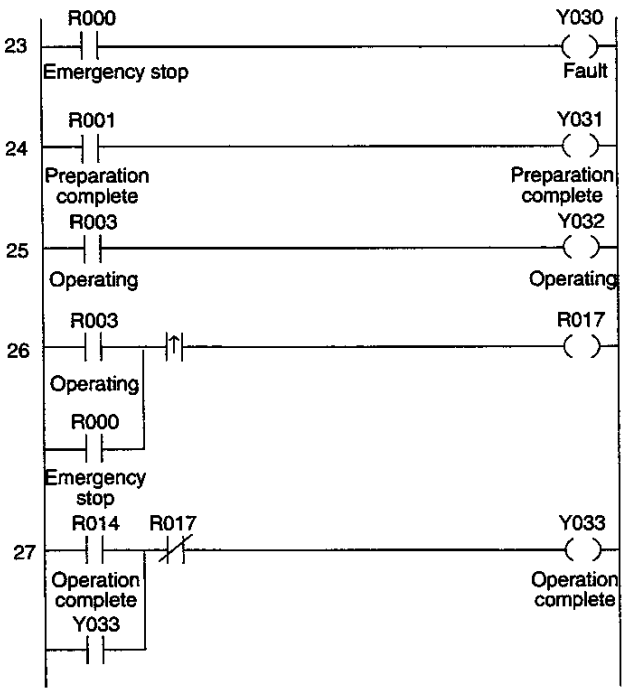
6. Programming Example



Stop Sequence



Lamp Circuit



When 'Emergency stop' (R000) is ON, the 'Fault' lamp (Y030) is put ON.

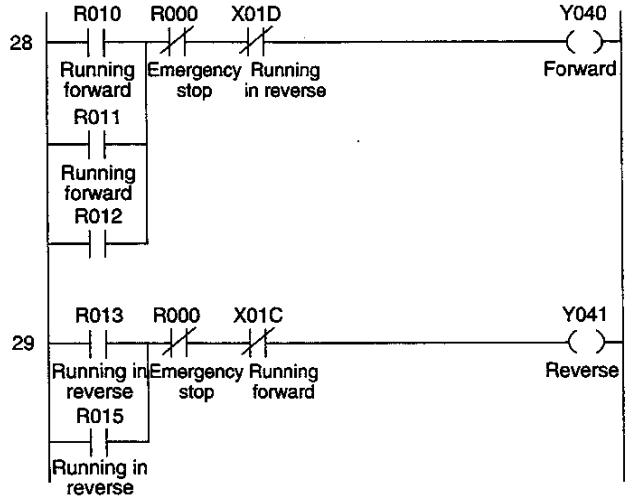
When 'Preparation complete' (R001) is ON, the 'Preparation complete' lamp (Y031) is put ON.

When 'Operating' (R003) is ON, the 'Operating' lamp (Y032) is put ON.

R017 becomes ON immediately the operation mode (R003) or 'Emergency stop' (R000) is put ON.

When 'Operation complete' (R014) is ON, the 'Operation complete' lamp (Y033) is put ON. When R017 is on, Y033 is reset.

Motor Circuit



When R010, R011, or R012 is ON, 'Forward' (Y040) is put ON. This is interlocked by the 'Running in reverse' answerback (X01D).

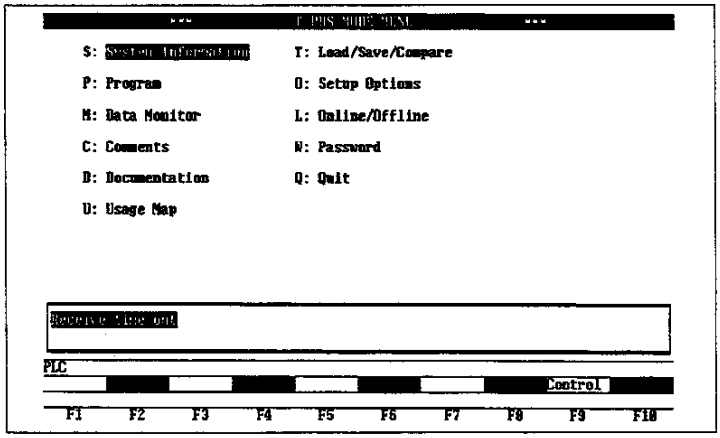
When R013 or R014 is ON, 'Reverse' (Y041) is put ON. This is interlocked by the 'Running forward' answerback (X01C).

Program End

6.4 Programming procedure

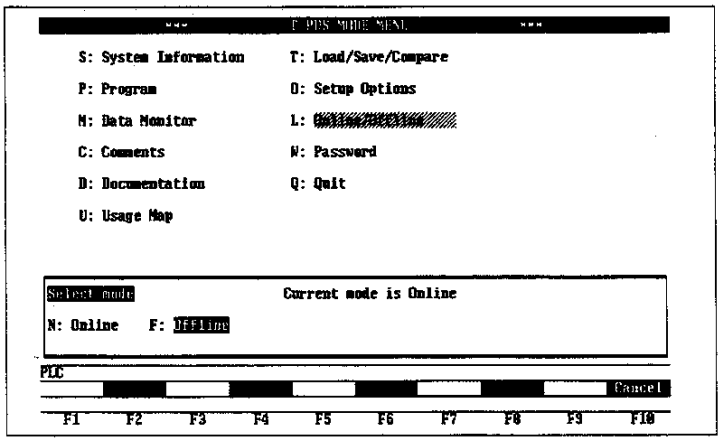
Here, the procedures for actually writing this program to the T3 using the programmer (T-PDS) are shown. (An operational example of T-PDS version 1.4)

(1) Turn the programmer power ON, startup the T-PDS by keying in TPDS [Enter].
(T-PDS initial Menu Screen)

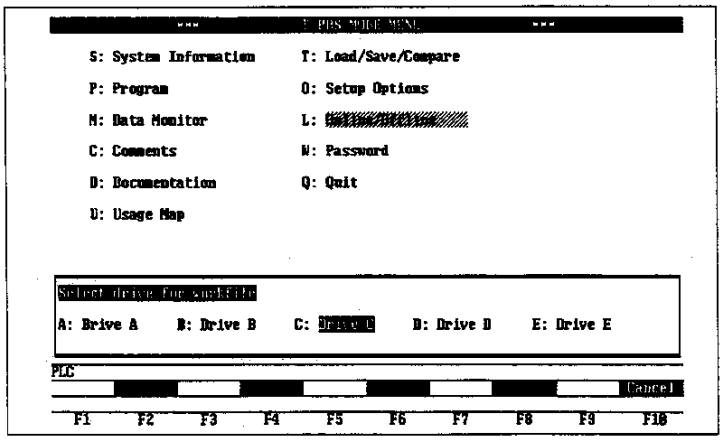


In the initial state, the T-PDS starts up in the communication mode with the PLC (T3). Therefore, in the state when it is not connected to the T3, "Receive time-out" is displayed on the screen.

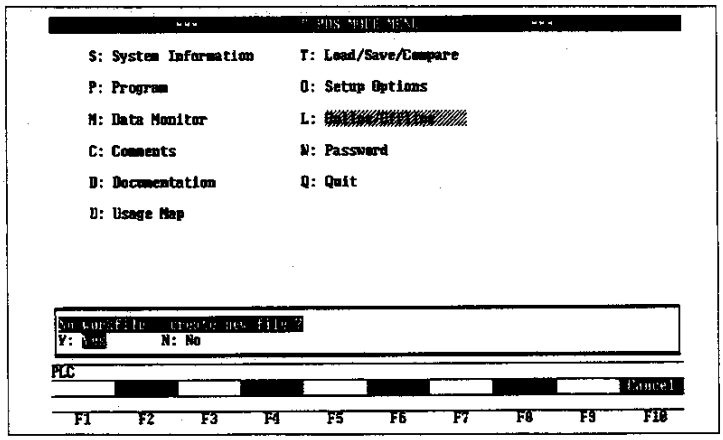
(2) In order to carry out off-line programming, change to off line mode. Select "L:Online/Offline", key-in L.



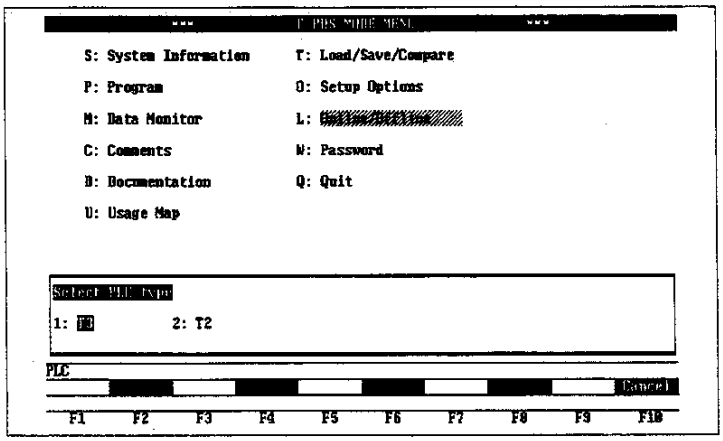
Here, select "F:Offline".
Key-in F.



This sets the selection mode for the disk drive in which to create the off-line work file. Here, select "C:drive C" by keying-in C.

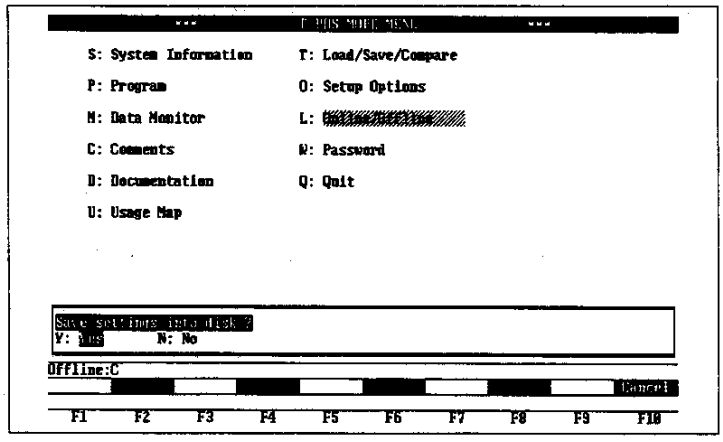


The programmer is now waiting for confirmation the creation of a work file. Key-in Y.



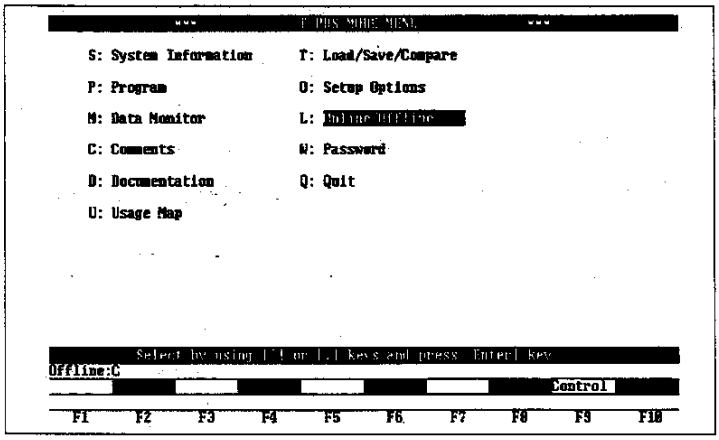
6. Programming Example

Next, the PLC model will be requested. Select "1: T3" by keying-in 1.

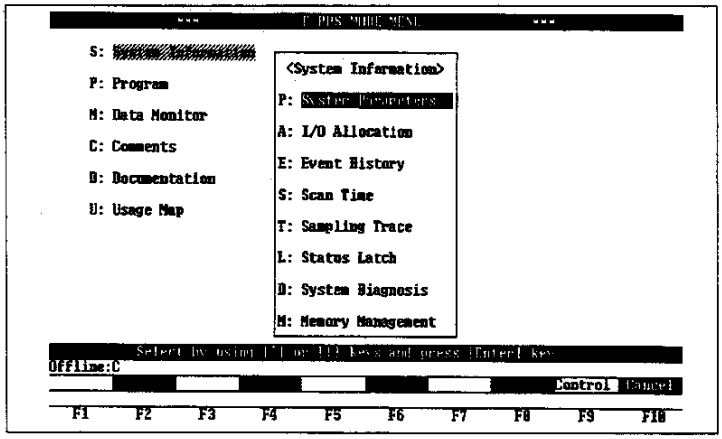


The T-PDS mode is changed to offline mode.
"Offline C: " is displayed at the bottom left of the screen.

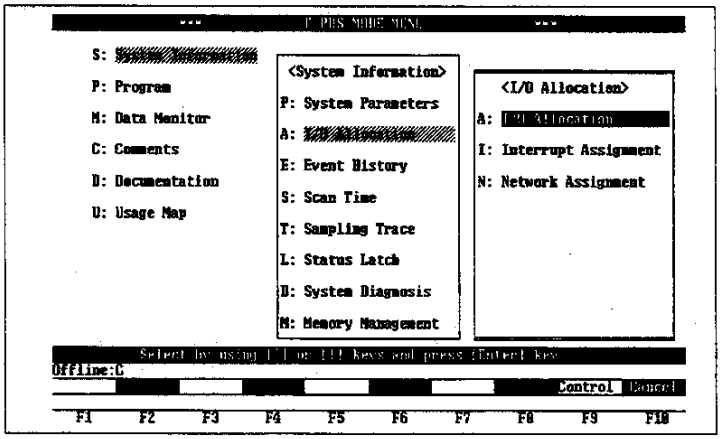
Since you are asked whether to record the settings, select "Y: Yes". By this means, the next time the T-PDS starts up, it will start up with the work file in drive C as the target.



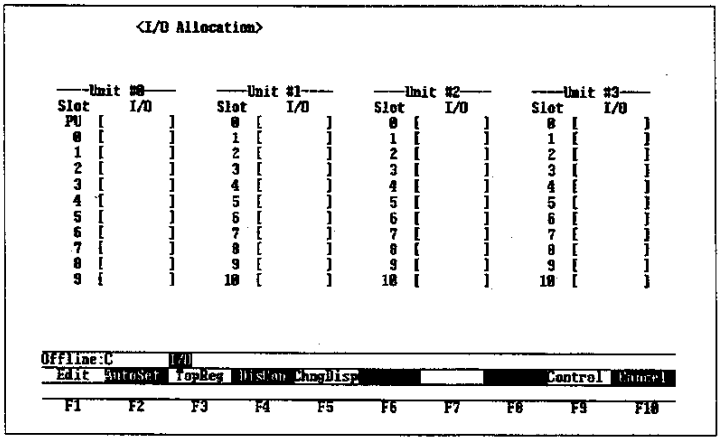
(3) Next carry out the I/O allocation. From the initial menu state, select "S: System information". Key-in S.



Here, select "A: I/O allocation information". Key-in A.

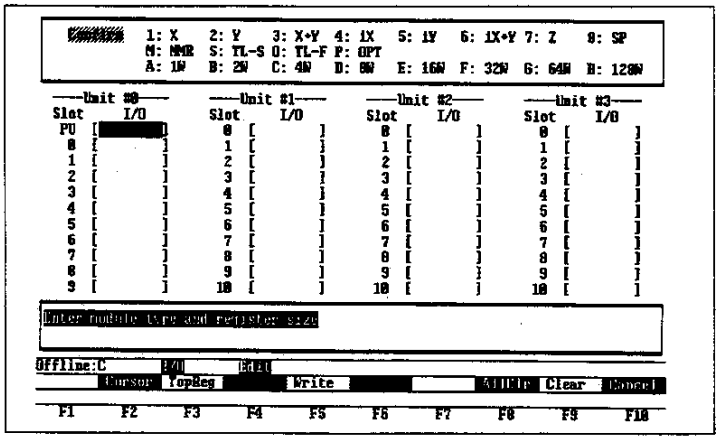


Then, key-in A to select "A: I/O allocation".



6. Programming Example

In offline programming, manual I/O allocation is carried out. Therefore, select F1 (Edit) on the command line. A cursor will appear on the screen.

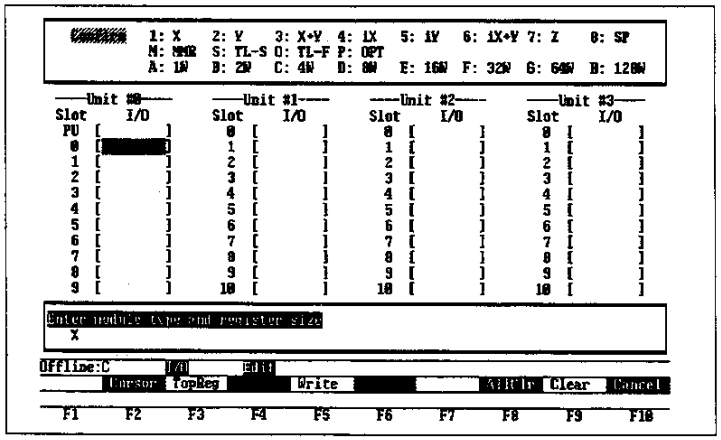

1: X 2: Y 3: X+Y 4: IX 5: IY 6: IX+Y 7: Z 8: SP
M: MMR S: TL-S O: TL-F P: OPT
A: 1W B: 2W C: 4W D: 8W E: 16W F: 32W G: 64W H: 128W
Unit #0
Unit #1
Unit #2
Unit #3
Slot I/O
PU
Offline:C
Edit
Cursor TopReg Write Clear Cancel
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10

Because the module configuration has been decided in Section 6.2, carry out the following settings as the input/output allocation.

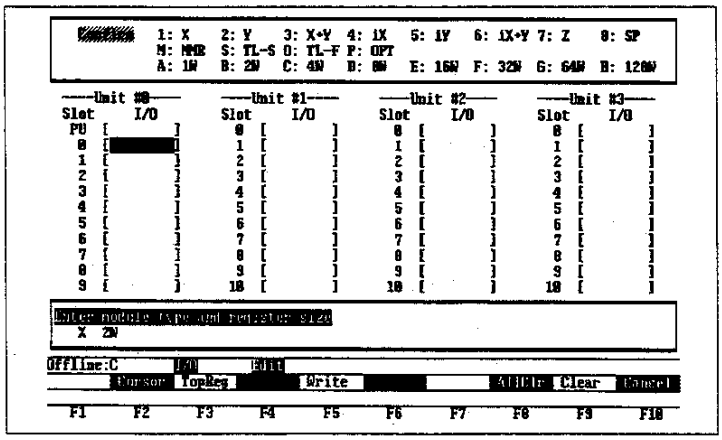
```
----- Unit #0 -----
  Slot      I/O
   PU     [       ]
    0     [ X 2W ]  ←  DI334(32 pts input)
    1     [ Y 2W ]  ←  DO334(32 pts output)
    2     [ Y 1W ]  ←  RO363S(16 pts output)
```

To set the module type, move the cursor to the specified slot position. Then designate by combining a function division (X, Y, etc) and the number of the register occupied (1W, 2W, etc) from the selection list displayed in the upper part of the screen.

First, move the cursor to Unit #0, Slot 0, using the cursor keys. Then, key-in 1 to designate "1: X".

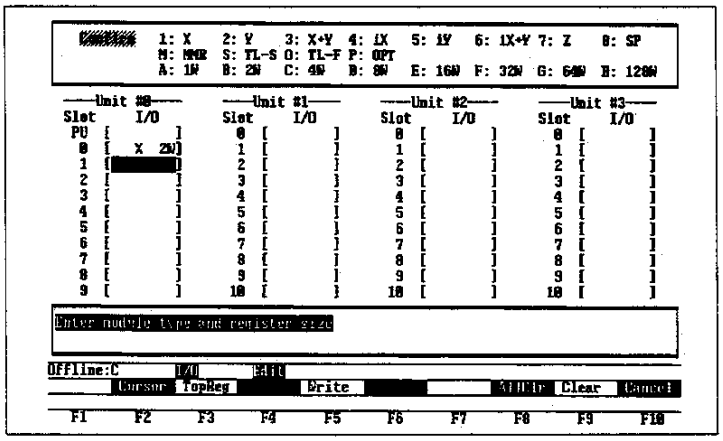

1: X 2: Y 3: X+Y 4: IX 5: IY 6: IX+Y 7: Z 8: SP
M: MMR S: TL-S O: TL-F P: OPT
A: 1W B: 2W C: 4W D: 8W E: 16W F: 32W G: 64W H: 128W
Unit #0
Unit #1
Unit #2
Unit #3
Slot I/O
PU
X
Offline:C
Edit
Cursor TopReg Write Clear Cancel
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10

Next, key-in B to designate "B: 2W".


1: X 2: Y 3: X+Y 4: iX 5: iY 6: iX+Y 7: Z 8: SP
M: MMR S: TL-S O: TL-F P: OPT
A: 1W B: 2W C: 4W D: 8W E: 16W F: 32W G: 64W H: 128W
Unit #0
Unit #1
Unit #2
Unit #3
Slot I/O
X 2W
Offline:C
Cursor TopReg Write Clear Cancel
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10

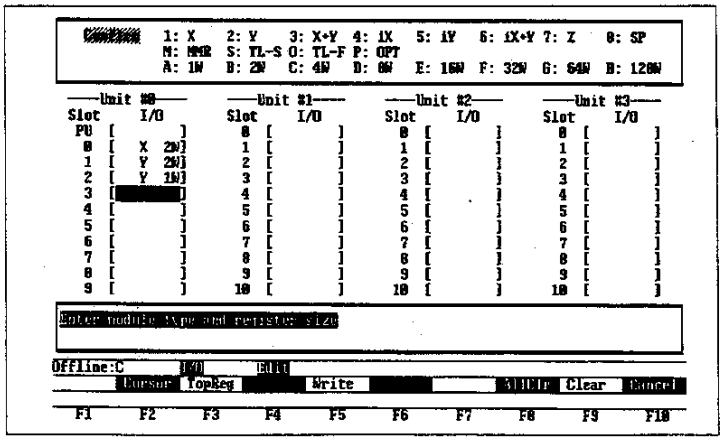
"X 2W" is displayed as the module type at the bottom of the screen.

Set the module type into the slot at the cursor position. Key-in [Enter].


1: X 2: Y 3: X+Y 4: iX 5: iY 6: iX+Y 7: Z 8: SP
M: MMR S: TL-S O: TL-F P: OPT
A: 1W B: 2W C: 4W D: 8W E: 16W F: 32W G: 64W H: 128W
Unit #0
Unit #1
Unit #2
Unit #3
Slot I/O
0 [X 2W]
Offline:C
Cursor TopReg Write Clear Cancel
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10

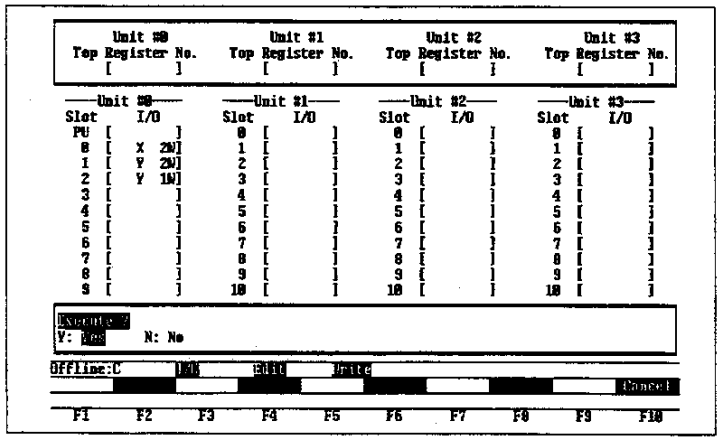
"X 2W" is displayed at the Unit #0, Slot 0 position, and the cursor moves to the next slot.

Hereafter, carry out the required settings using the same procedure.

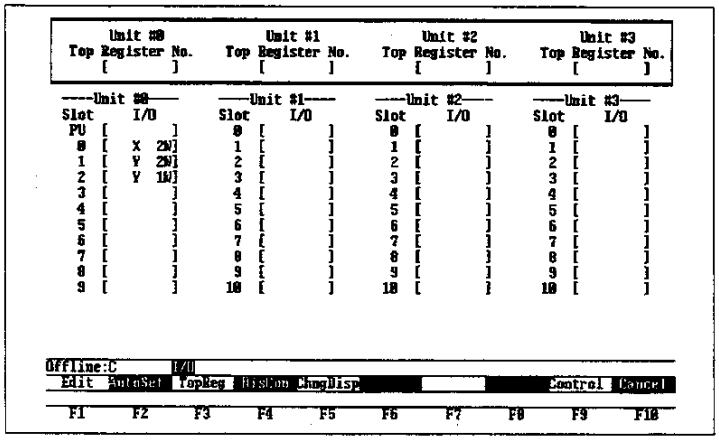

1: X 2: Y 3: X+Y 4: iX 5: iY 6: iX+Y 7: Z 8: SP
M: MMR S: TL-S O: TL-F P: OPT
A: 1W B: 2W C: 4W D: 8W E: 16W F: 32W G: 64W H: 128W
Unit #0
Unit #1
Unit #2
Unit #3
Slot I/O
0 [X 2W]
1 [Y 2W]
2 [Y 1W]
Offline:C
Cursor TopReg Write Clear Cancel
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10

6. Programming Example

After completing required settings, write this information in the work file. Select F5 (Write) from the command line.

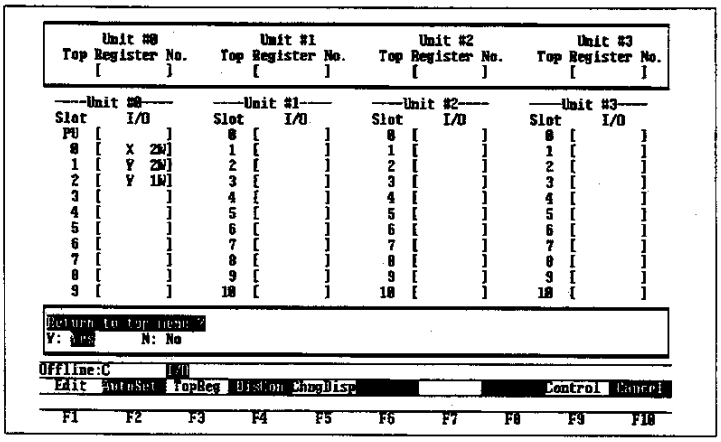


The programmer will wait for confirmation. Key-in Y.

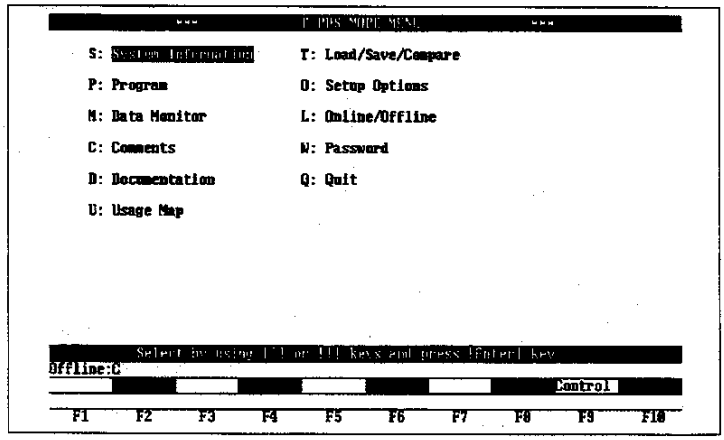


This completes the I/O allocation.

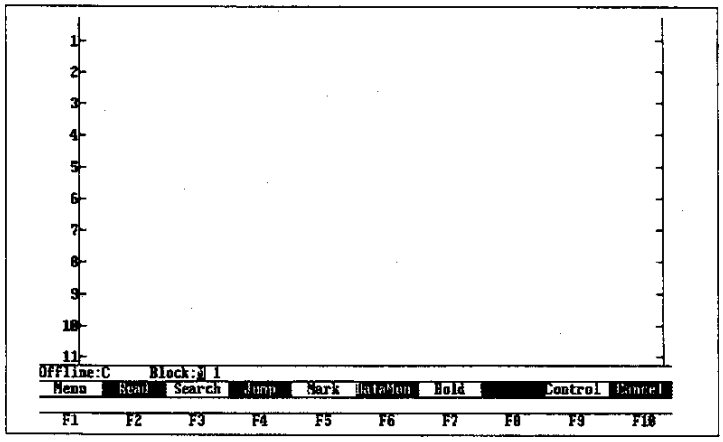
(4) Now we enter the program typing phase. First, press the [Esc] key to return to the initial menu.



The programmer will wait for confirmation. Key-in Y.



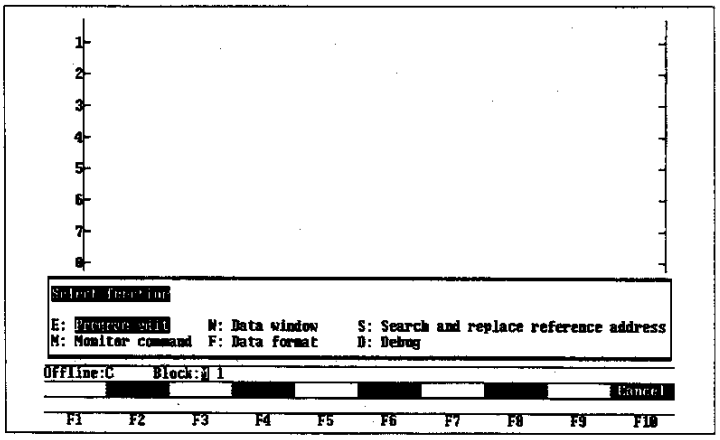
Here, select "P: Program". Key-in P.



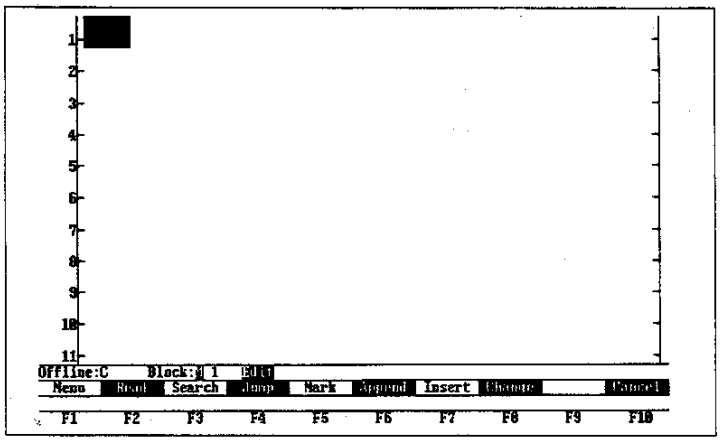
6. Programming Example

Block 1 of the main program is automatically selected, and "Block: M1" will be displayed on the screen.

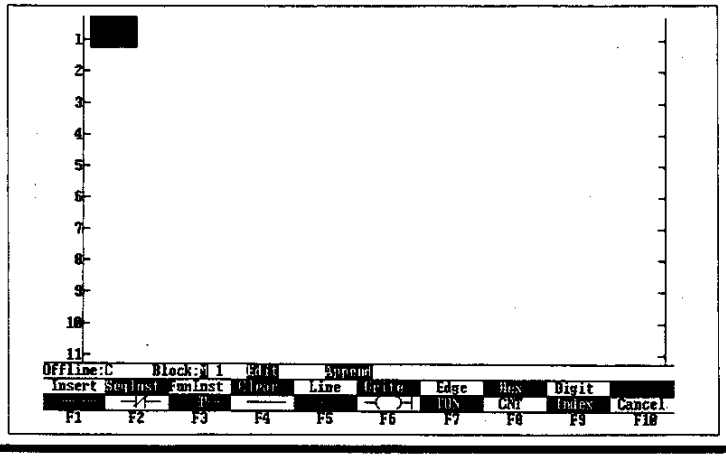
Here, select F1 (Menu) from the command line.



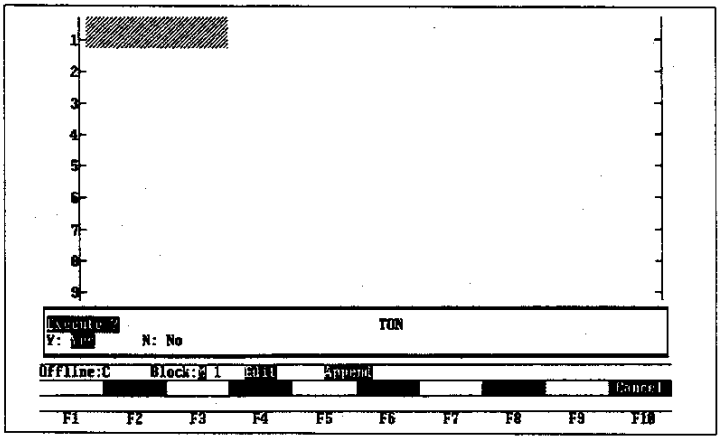
Then, select "E: Program Edit" from the menu window and key-in E. A cursor will appear on the screen.



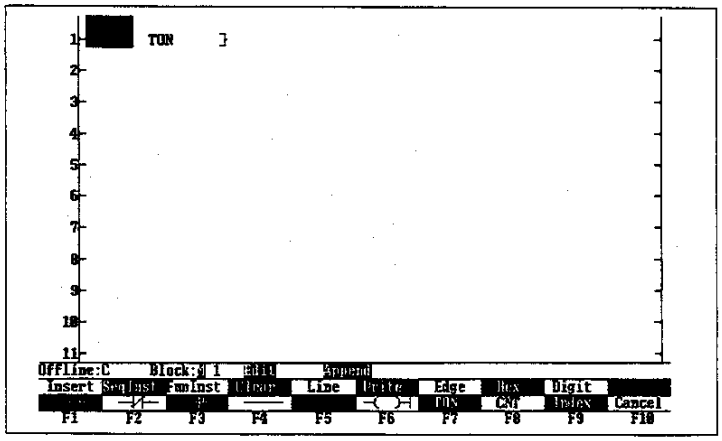
Here, select F6(Append) from the command line. Then, instruction symbols will appear on the command line.



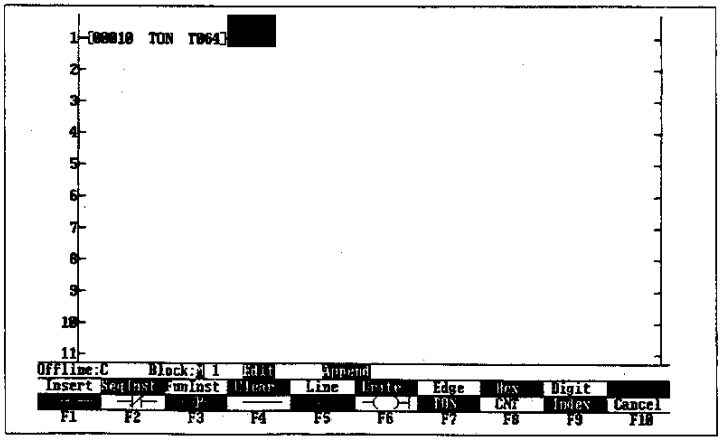
Here, start typing in the program in Section 6.3.
First, press F7 (TON).



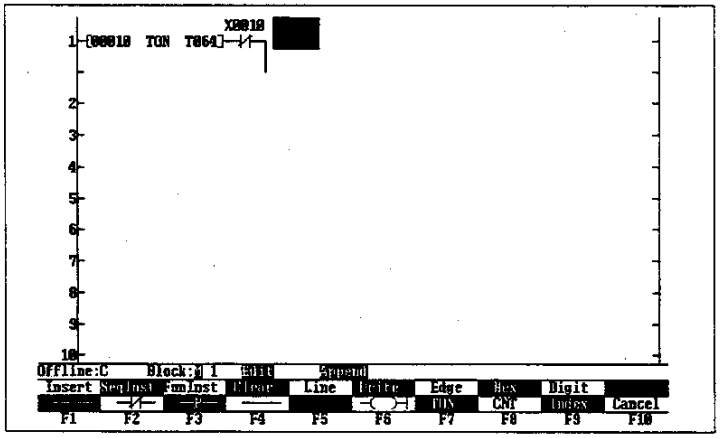
Since the programmer is waiting confirmation, key-in Y.



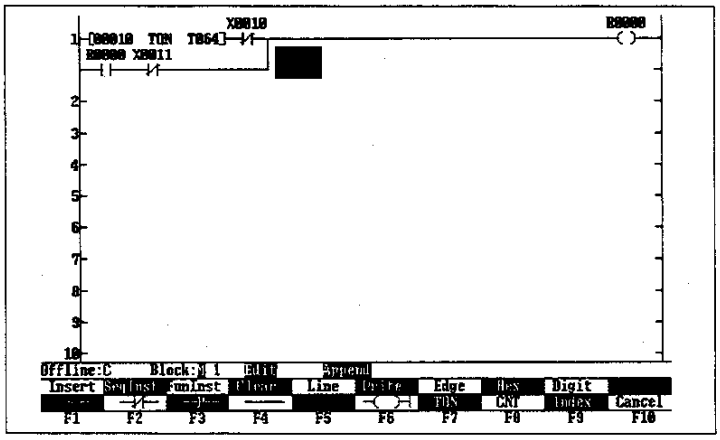
Input the operands (setting value and timer register).
Key-in
10 [Enter]
T64 [Enter]
If you make a mistake, cancel it with the Space key and re-input.



Next, input the NC contact of X010 (with vertical connection).
Key-in
F2 (⊣/⊢)
F5 (|)
X10 [Enter]



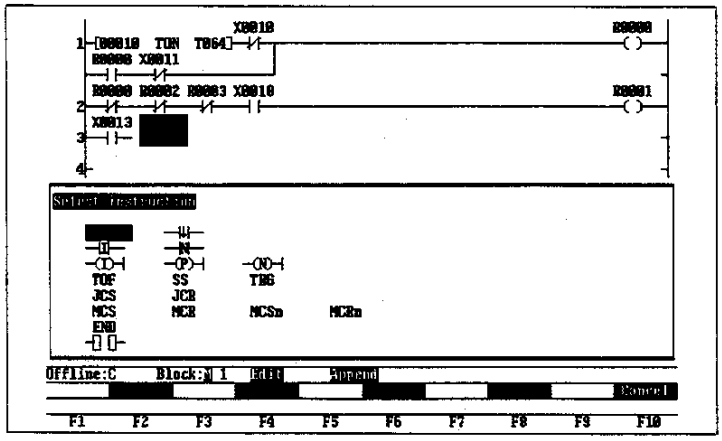
Complete the 1st rung using the same procedure, as follows.
Key-in
F6 (⊣ ⊢) R0 [Enter]
F1 (⊣ ⊢) R0 [Enter]
F2 (⊣/⊢) X11 [Enter]
F4 (-) [Enter]
F4 (-) [Enter]



Next, move to the head of the 2nd rung using the cursor keys, and input the 2nd rung using the same procedure as for the 1st rung. Then input the 3rd and 4th rungs.

While doing this, when an instruction for which the symbol is not displayed on the command line such as a transitional contact, display the Menu Window by pressing Shift +F2 (Seq Inst), and then select.

(Screen state when Shift +F2 (Seq Inst) has been pressed)



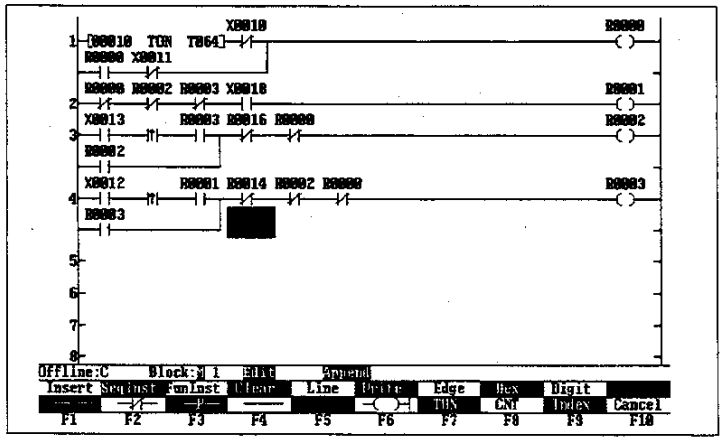
When input has been completed as far as the 4th rung, write the 1st to 4th rungs into the work file.

In other words, the size of program which is writable to the work file at any one time is one screen size (11 lines by 12 columns). Therefore, this means writing to the work file at convenient divisions of rungs.

(The cursor also will move within the screen limits in the Edit mode).

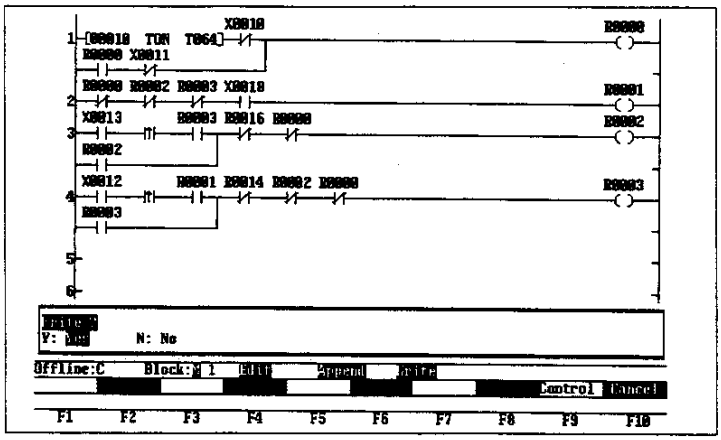
Carry out the operation of writing to the work file as follows:

(State with input up to the 4th rung complete)

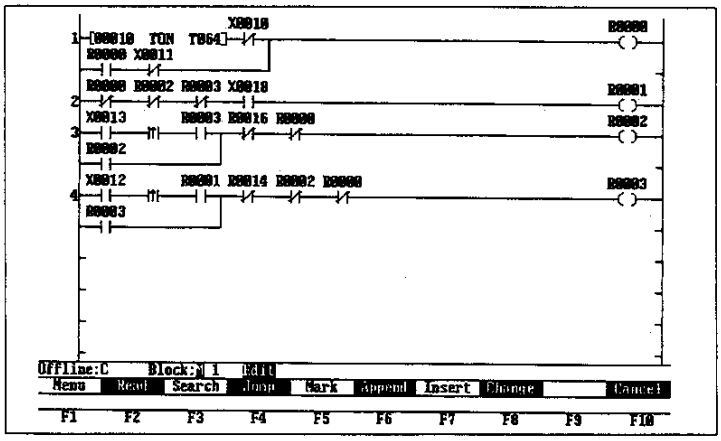


6. Programming Example

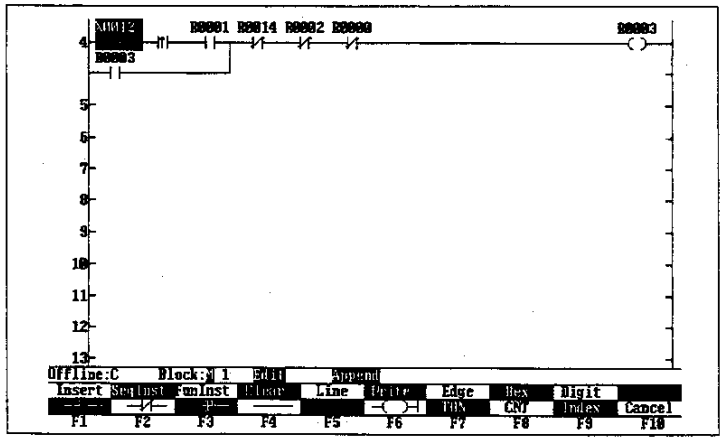
Key-in Shift + F6 (Write).



Because the programmer is awaiting confirmation, key-in Y.



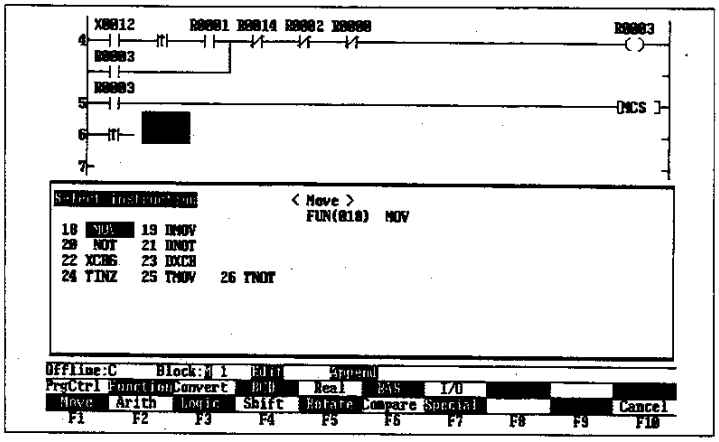
In this way, the 1st to 4th rungs have been written to the work file. Next, input the 5th rung onward. Move the cursor onto the 4th rung using the cursor keys (the cursor can move over existing rungs only), and press F6 (Append). The screen will then turn to an edit screen with the 4th rung leading.



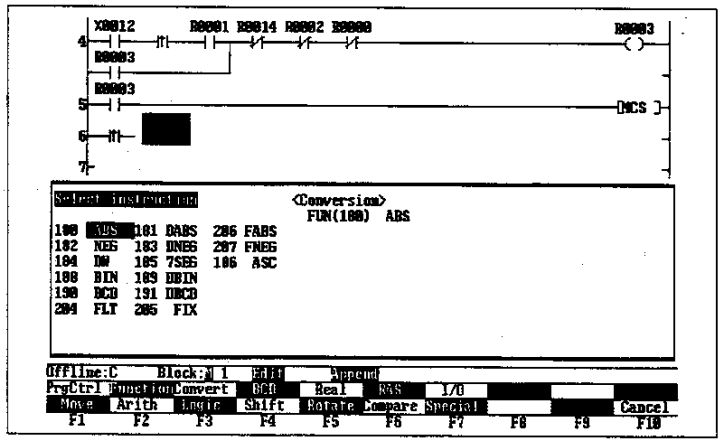
Move the cursor to the head of the 5th rung, and start entering the program from there onward.

To input the Function instructions such as BIN, key-in Shift+F3 (Fun Inst) to display Function instructions, then select the instruction group which contains the desired instruction.

(Screen state when Shift+F3 (Fun Inst) has been pressed)


X0012
R0003
MCS
< Move >
FUN(010) MOV
18 MOV 19 DMOV
20 NOT 21 DNOT
22 XCHG 23 DXCH
24 TINZ 25 TMOV 26 TNOT
Offline:C Block:# 1 Edit Append
PrgCtrl Function Convert BCD Real BAS I/O
Move Arith Logic Shift Rotate Compare Special Cancel
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10

(State when Shift+F3 (Convert) is pressed on the above screen)


X0012
R0003
MCS
<Conversion>
FUN(100) ABS
100 ABS 101 DABS 206 FABS
102 NEG 103 DNEG 207 FNEG
104 DN 105 7SEG 106 ASC
108 BIN 109 DBIN
190 BCD 191 DBCD
204 FLT 205 FIX
Offline:C Block:# 1 Edit Append
PrgCtrl Function Convert BCD Real BAS I/O
Move Arith Logic Shift Rotate Compare Special Cancel
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10

Hereafter, write the whole program using the same procedure.

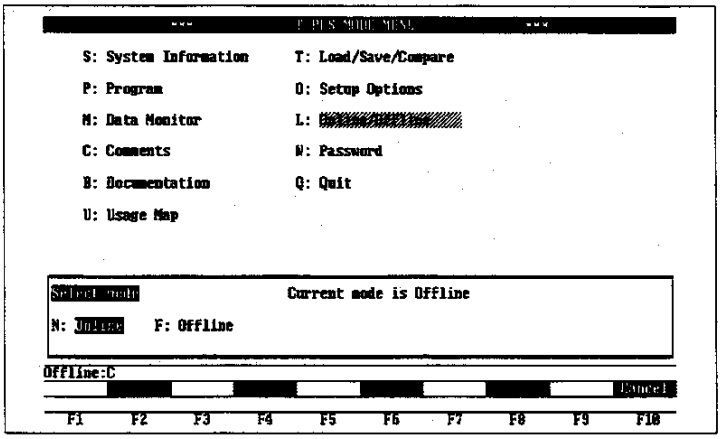
(5) When the whole program has been written in work file in the operation up to this point, load the program into the T3.

First, connect the T3 and the T-PDS with the dedicated cable. (This assumes that the modules in Section 6.2 are mounted in the T3)

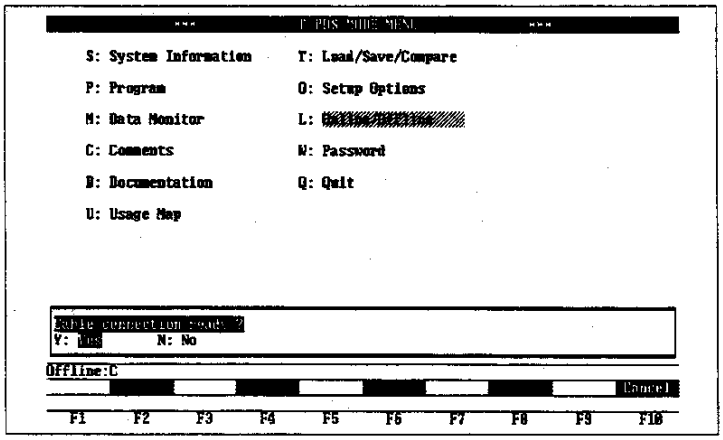
Next, put the RAM/ROM switch on the CPU to RAM, the operation mode switch to the RUN position, and turn on power to the T3. (The T3 will start up in the HALT mode).

(6) Put the T-PDS into communication mode with the PLC (T3).

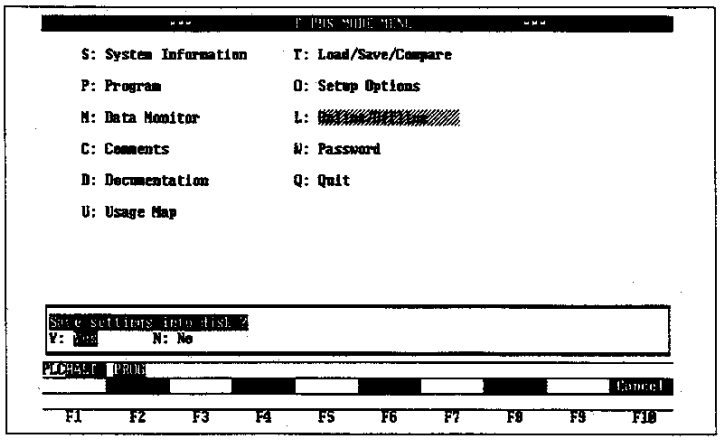
First, return the T-PDS display to the initial menu by pressing [Esc] [Enter], and then select "L:Online/Offline".



Here, select "N: Online" to select online mode. Key-in N.



Confirm the connection state and key-in Y

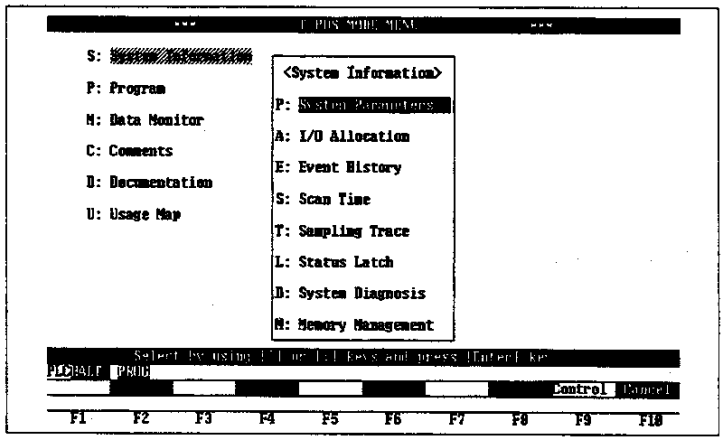


When the communication of the T-PDS with the T3 is correctly connected, "PLC HALT" message will be displayed at the bottom left of the screen.

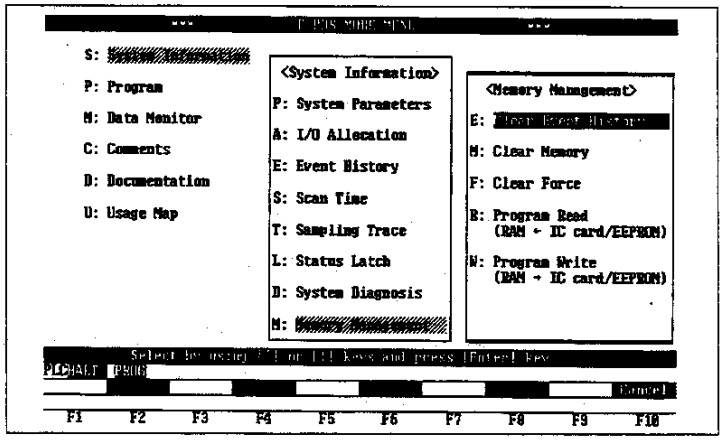
Although the record of settings is not always required, here, select "Y: Yes".

Now the T-PDS has been changed to the online mode.

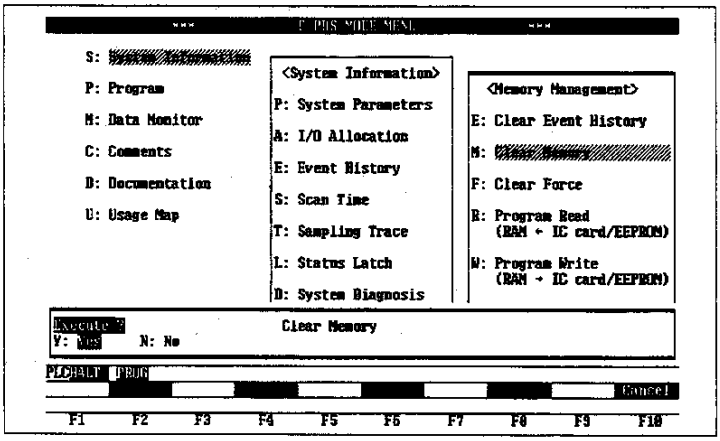
(7) Next, clear the memory of the T3. Select "S:System Information". Key-in S.



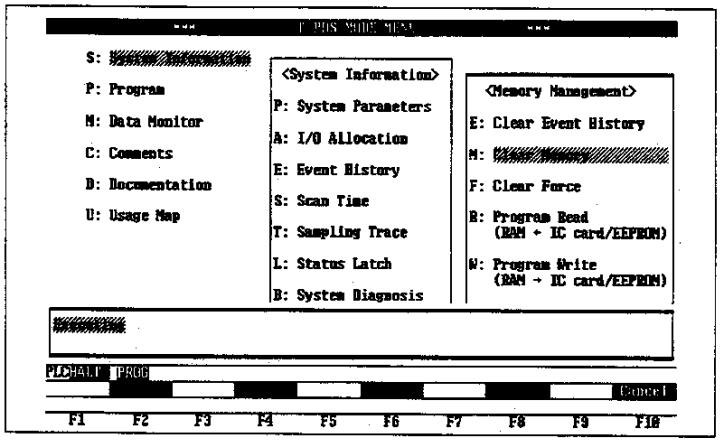
Here, select "M: Memory Management" from the <System Information> menu. Key-in M.



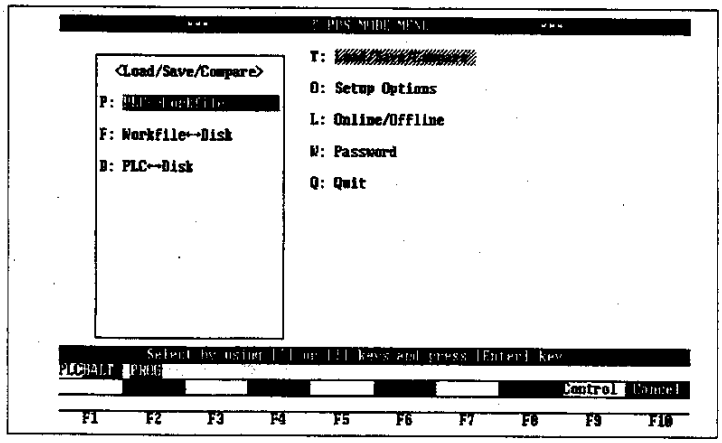
Next, select "M: Clear Memory" from the <Memory Management> menu. Key-in M.



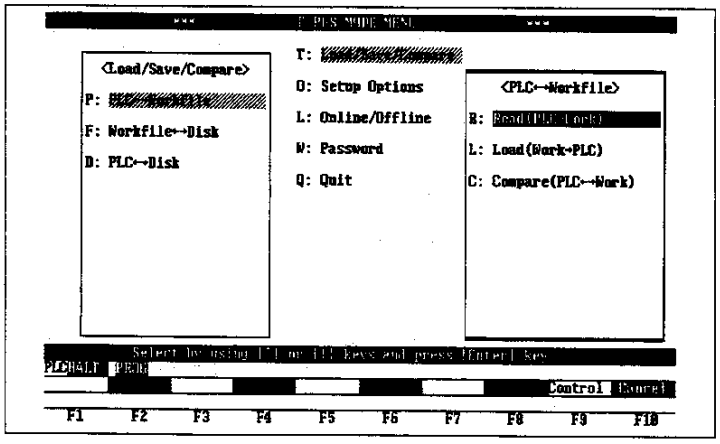
The programmer will await execution confirmation. Execute Clear by keying-in Y.



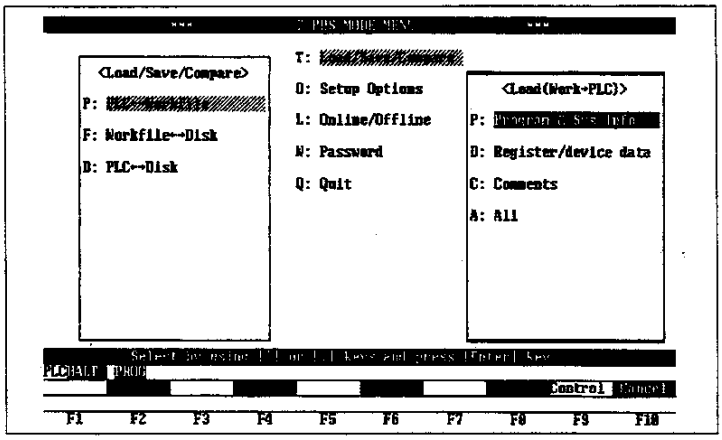
(8) Next, transfer (load) the program which has been written in the work file to the T3. First, display the initial menu by pressing [Esc][Enter], and then select "T: Load/Save/Compare".



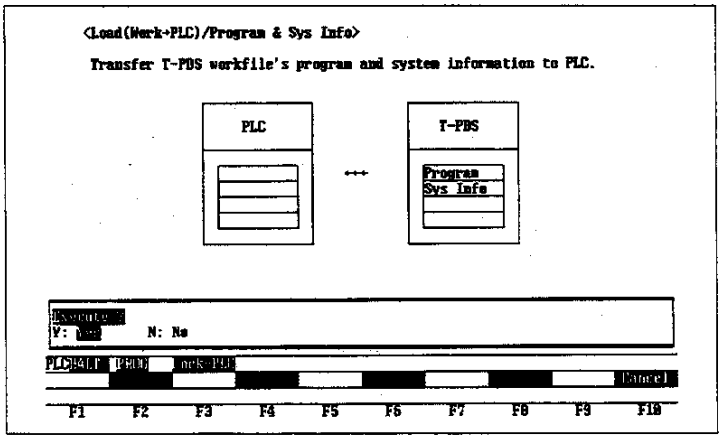
Here, select "P: PLC ↔ Work File" from the <Load/Save/Compare> menu.



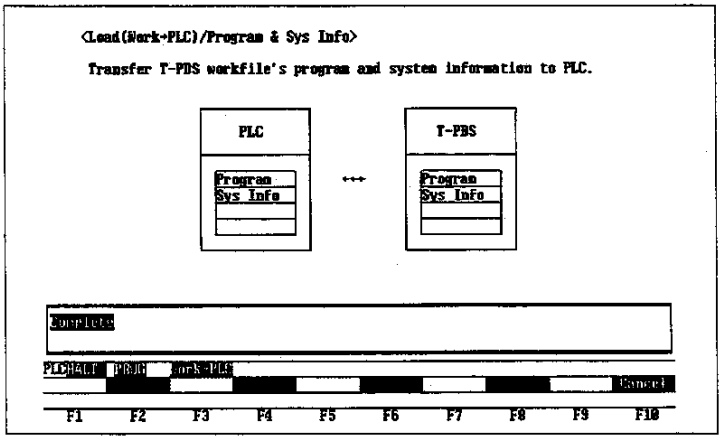
Then, select "L: Load (Work → PLC)" from the <PLC ↔ Work File> menu.



The selection menu for loading details is displayed. Since it is simply the program in this example, select "P: Program & Sys Info".

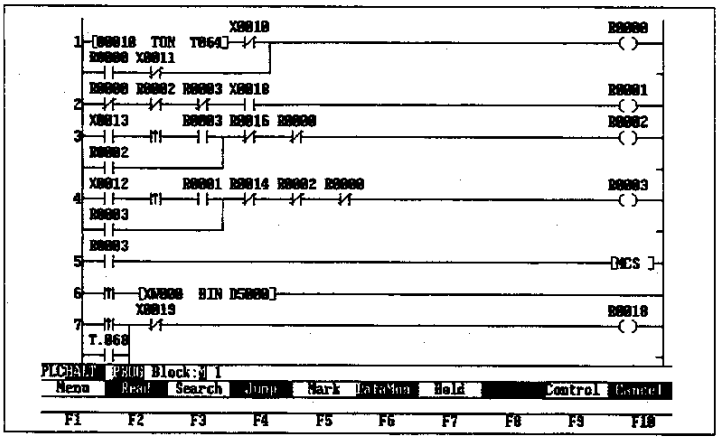


The programmer will await execution confirmation. Key-in Y.



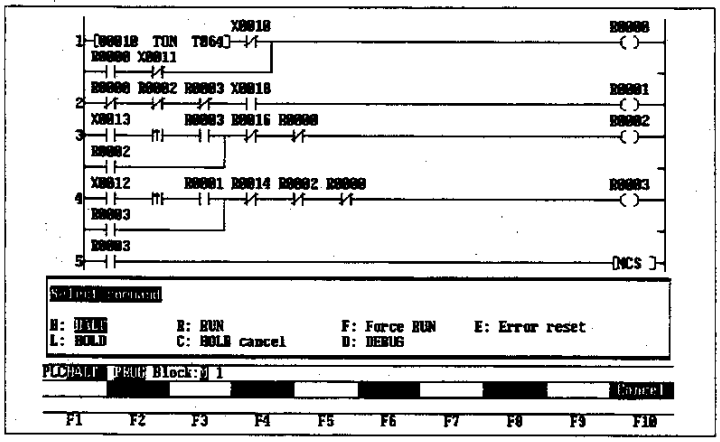
When correct loading has been carried out "Complete" Will be displayed.

(9) When the loading of the program has been completed by the above operations, operate the T3 (RUN mode) and debug the program. Here, try to change the T3 mode by the Control command of the T-PDS. First display the initial menu by pressing [Esc] [Enter], and then select "P:Program".



The T-PDS will enter the monitor mode for the program which has been loaded in the T3.

Here, select F9 (Control) from the command line.

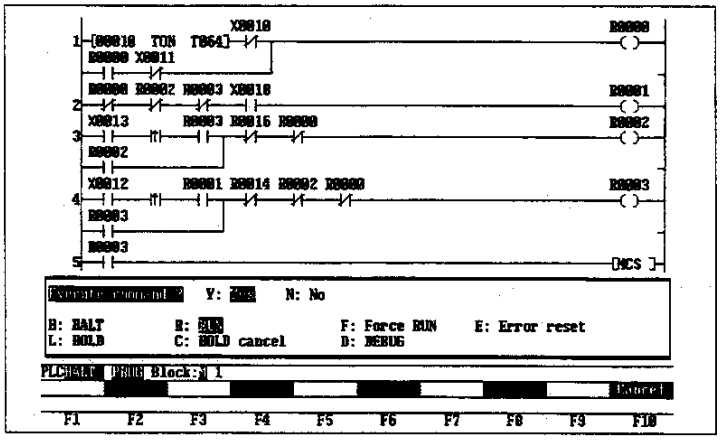


NOTE

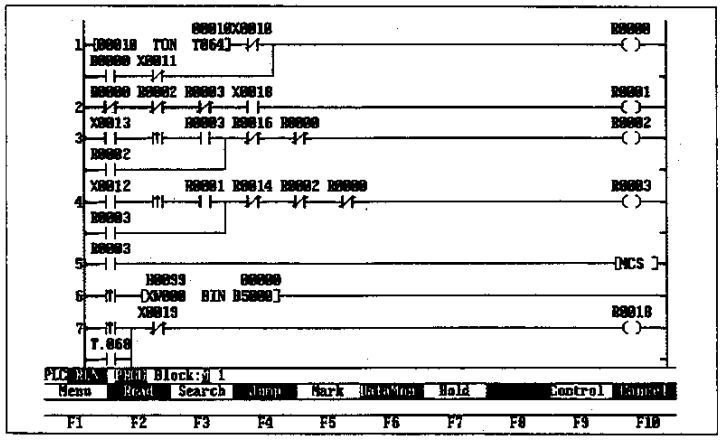
When the T3 is put into the RUN mode with the aim of program debugging and test running, take thorough precautions for safety, such as switching OFF the motive power circuit.

6. Programming Example

Select "R: RUN" from the menu window.



The programmer will await execution confirmation, Key-in Y after re-checking the safety of the surroundings.



"PLC: RUN" will be displayed on the screen. This is the monitor screen for the program execution state.

Perform confirmation of operation by using the external simulation switch and the T-PDS simulation input function (Force function). For operation, see separate T-PDS operation manual.

When carrying out program correction/modification, stop the T3 temporarily (put into the HALT mode), and correct/modify the program in the T3.

When carrying out creation/modification of the program while still in the online mode, the operations are the same as in the offline mode.

(10) When program correction and operation check are completed, save the program in the disk and switch OFF the T3 power.

To finish with the T-PDS, press [Esc] [Enter] and select "Q: Quit" in the state with the initial menu displayed.

The above completes the programming procedure. If the T3's RAM/ROM switch is put to ROM and the Operation Mode switch is put to RUN (or P-RUN), the T3 will operate automatically when power is next switched ON.

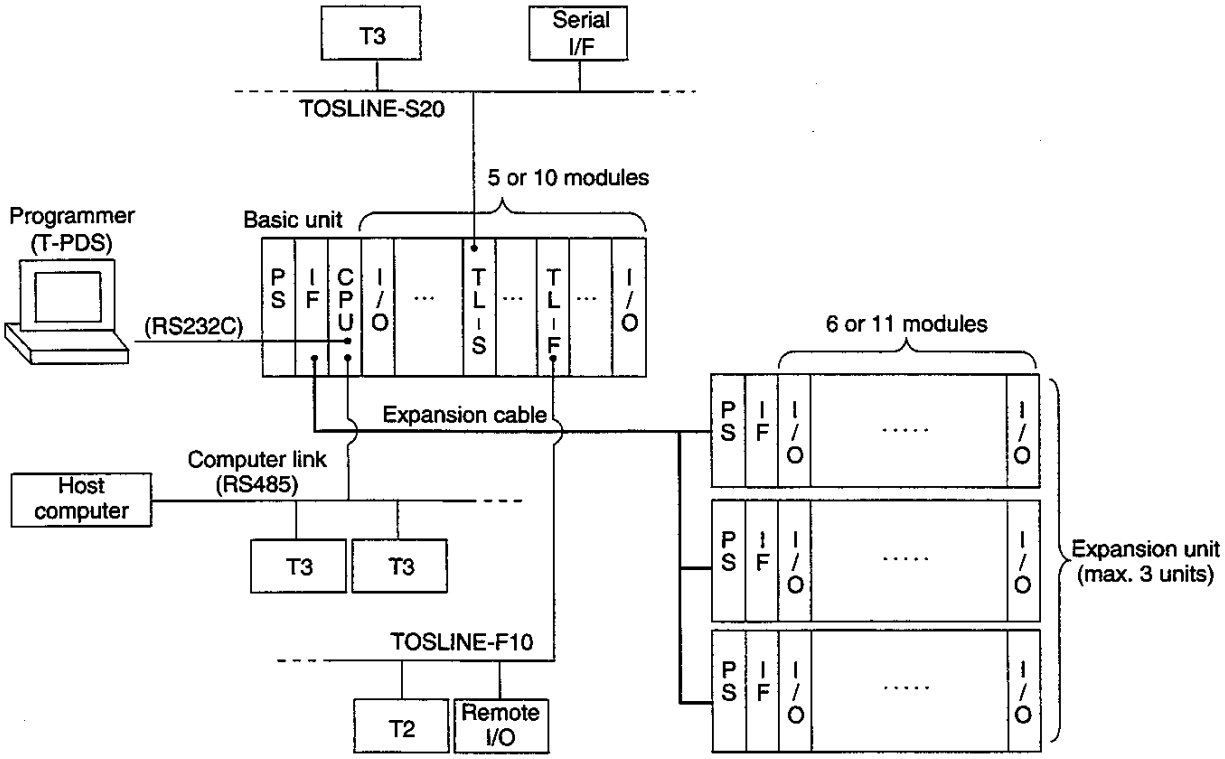
NOTE

In the case of a CPU with a built-in EEPROM (PU325), write the program into the EEPROM before the above procedure (10). The operation can be performed by selecting "W: Program Write" from the <Memory Management> menu. (See the screen on the procedure (7)).

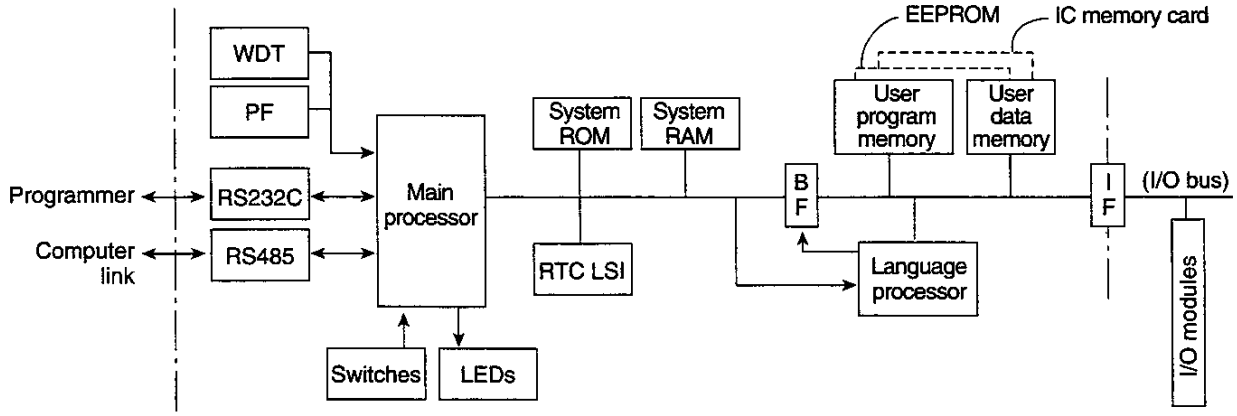
PART 2
FUNCTIONS

1.1 T3 System configuration

The T3 system configuration is shown in the figure below. Part 2 explains the T3 system functions, concentrating on the T3 CPU functions.



The internal block diagram of the T3 CPU is shown below.



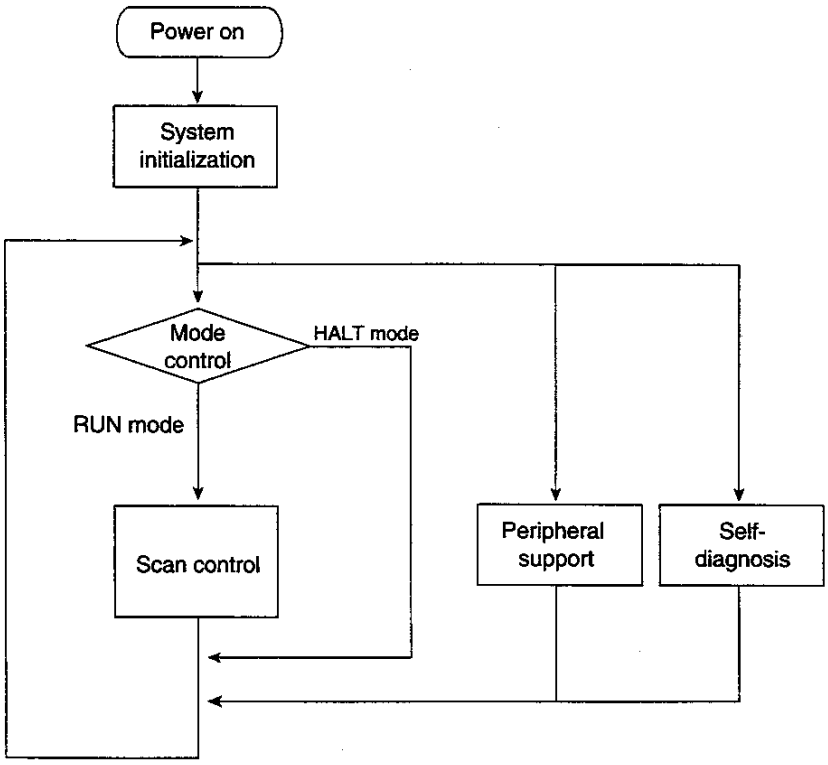
The Main processor controls overall execution tasks. The Language processor (LP) works as co-processor and executes the user program (bit operation and word operation). These two processors work in parallel during scan operation.

1.2 Functional specifications

Item		Specification
Control method		Stored program, cyclic scan system
I/O method		Batch I/O (refresh), Direct I/O, or combination
Number of I/O points		1376 points (when 32 pts I/Os are used) 2752 points (when 64 pts I/Os are used) Total space: 4096 points/256 words
User Program	Programming language	SFC (Sequential Function Chart) Ladder diagram (relay symbol+function block)
	Program capacity	32k-steps (incl. comment space) (1 step=24bits)
	Memory	Main memory: SRAM (battery back up) Optional memory: EEPROM/IC memory card
	Instructions	Basic ladder instructions: 24, function instructions:201 * transfer (single length/double length/register table) * arithmetic calculation (single length/double length/binary/BCD) * logical operation (single length/double length/register table/bit file) * comparison (single length/double length, sign/unsign) * program control (jump/FOR-NEXT/subroutine and others) * function (limit/trigonometric integral/PID/function generator) * conversion (ASCII/BCD/7 segment other) * Floating point operations
	Execution speed	0.15 µs/contact, 0.3 µs/coil 0.9 µs/transfer, 2.25 µs/addition
Scanning system		Floating scan/constant scan (interval: 10-200 msec. 10 msec units)
Multitasking		1 main program, 4 sub-programs 1 timer interrupt (2-1000 msec, 1 msec units), 8 I/O interrupt
User data	I/O device/register	4096 points/256 words (X/Y, XW/YW, batch I/O) (I/O, IW/OW direct I/O)
	Auxiliary device/register	8192 points/512 words (R/RW)
	Special device/register	4096 points/256 words (S/SW)
	Timer device/register	512 points (T./T) (T000-T063: 0.01 sec) (T064-T511: 0.1 sec)
	Counter device/register	512 points (C./C)
	Data register	8192 words (D)
	Link device/register	8192 points/1024 words (Z/W) (for TOSLINE-S20)
	Link relay/register	4096 points/256 words (L/LW) (for TOSLINE-F10)
	File register	8192 words (F)
	Index register	I, J, K (total 3 words)
	Retentive memory	User specified for RW, T, C and D
RAS	Diagnosis	Battery level, I/O bus check, I/O response, I/O registration, I/O parity, Watch dog timer, illegal instruction, LP check, others
	Monitoring	Event history record, scantime measument, others
	Debugging	Online trace monitor, force, sampling trace, status latch, single step/N scan execution, break point, others

2.1 Basic internal operation flow

The T3 basic operation flow chart is shown below.



T3 performs system initialization following power on. If no abnormality is detected, T3 proceeds the mode control processing.

Here, if the RUN mode transitional condition is fulfilled, the scan control begins. The scan control is the basic function of the T3 for the user program execution operation. And if the RUN mode transitional condition is not fulfilled, T3 enters the HALT mode and does not execute the user program.

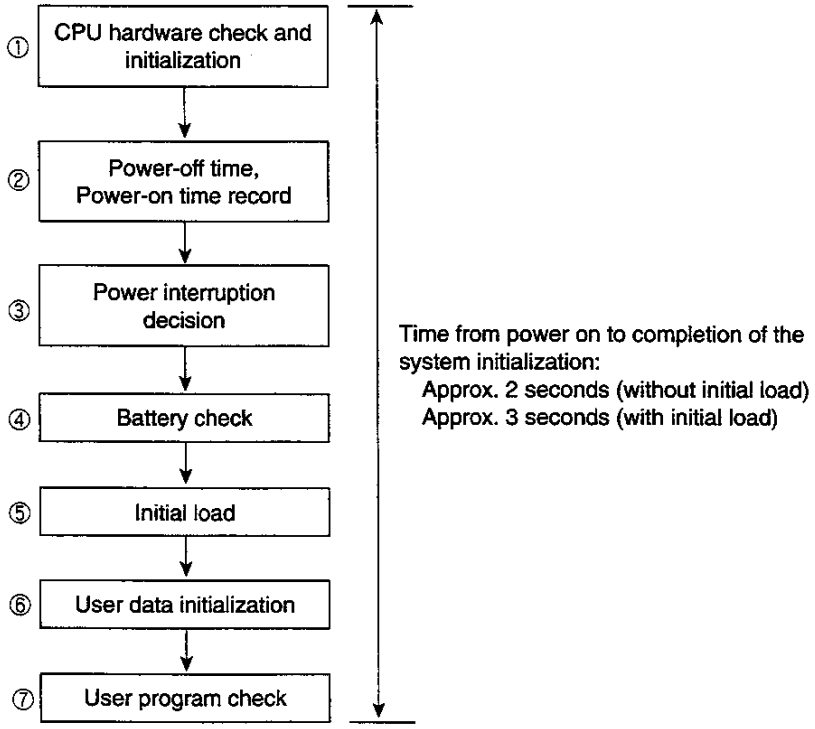
The peripheral support processing is executed as background for communicating with the programmer and the computer link.

Self-diagnosis is carried out in each processing. The above figure shows the self diagnosis executed as background.

The details of these processes are explained in this section. Self-diagnosis is explained in 5 RAS functions.

2.2 System initialization

The system initialization is performed after power is turned on. The following flow chart shows the sequence of processes explained below.


① CPU hardware check and initialization
② Power-off time, Power-on time record
③ Power interruption decision
④ Battery check
⑤ Initial load
⑥ User data initialization
⑦ User program check
Time from power on to completion of the system initialization:
Approx. 2 seconds (without initial load)
Approx. 3 seconds (with initial load)

① CPU hardware check and initialization
System ROM check, system RAM check and initial set up, peripheral LSI check and initial set up, RTC LSI check, and language processor (LP) check take place.

② Power-off time, Power-on time record
The last time the power was switched off is recorded in the event history table, and the present date and time read from the RTC LSI is recorded as power-on time. Also the present date and time are set into the special register (SW007-SW013).

③ Power interruption decision
In the hot restart mode (S0400 is ON), if power-off period is less than 2 seconds, it is decided as power interruption. In this case, initial load and user data initialization explained below will not be carried out. (only when the last power-off occurred in the RUN mode)

④ Battery check
The battery voltage is checked for the user program and the user data backup. If the battery voltage is lower than the specified value a message is recorded in the event history table 'batt voltage drop' together with the special relay battery alarm flag (S000F) setting.

⑤ Initial load

The initial load means the term for the transfer of the contents of the user program and the leading 4k words of the data register (D0000 to D4095), from the peripheral memory (IC memory card or EEPROM) to the main memory (RAM), prior to running the user program.

The initial load is initiated when the power is turned on, the operation mode switch is in other than P-RUN and the RAM/ROM switch is turned to ROM.

* The initial load is not performed if both the IC memory card and the EEPROM are not present.

* When the IC memory card and the EEPROM are both present, the IC memory card will become the transfer source.

* The initial load is not performed if the user program is written in the IC memory card or the EEPROM, but the contents are destroyed (BCC error detection).

* When the user program is not written in the IC memory card, the EEPROM will become the transfer source.

⑥ User data initialization

The user data (registers and devices) is initialized according to the conditions in the following table:

Register/Device	Initialization
Input registers/devices(XW/X)	For forced input devices, the previous state is maintained, the others are 0-cleared.
Output registers/devices(YW/Y)	For coil forced output devices, the previous state is maintained, the others are 0-cleared.
Auxiliary registers/devices (RW/R)	For registers designated as retentive and coil forced devices, the previous state is maintained, the others are 0-cleared.
Special registers/devices (SW/S)	CPU setting part is initialized and the user setting part is maintained.
Timer registers/relays (T/T.)	For registers designated as retentive and the device corresponding to the previous state is maintained, the others are 0-cleared.
Counter registers/realys (C/C.)	
Data registers (D)	For registers designated as retentive, the previous state is maintained, the others are 0-cleared. If the Operation Mode switch is in P-RUN, leading 4k words (D0000 to D4095) are maintained.
Link registers/relays (W/Z)	For forced link devices the previous state is maintained, the others are 0-cleared.
Link relays (LW/L)	For forced link relays, the previous state is maintained, the others are 0-cleared.
File registers (F)	All maintained
Index registers (I, J K)	All 0-cleared

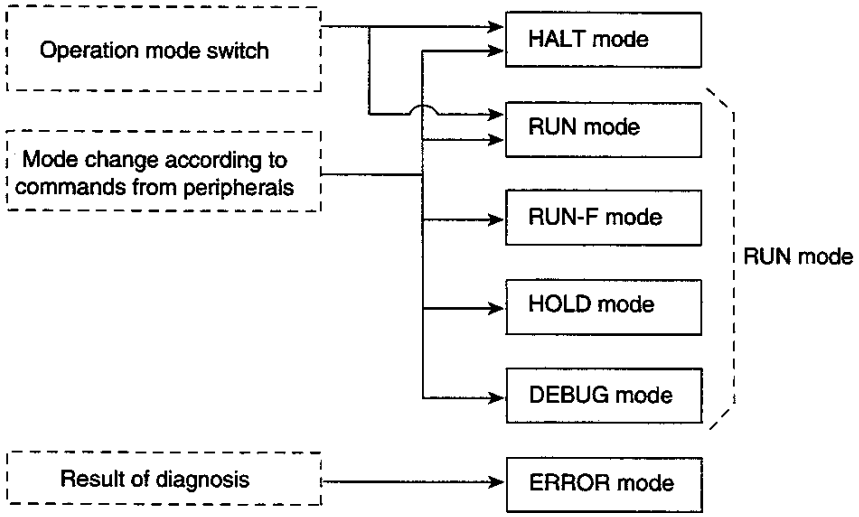
*1) For the force function, refer to 5.11 Debug Support Function.
*2) For the retentive memory area designation, refer to Part 3, Section 2.2.

⑦ User program check
The contents of the user program on the main memory (RAM) are checked by BCC.

2.3 Mode control

The T3 operation mode is selected according to the status of the operation mode switch on the CPU module and mode change requests from the peripherals (programmer, computer link, data transmission system).

The T3 operation mode is basically divided into three; RUN mode, HALT mode and ERROR mode. Also, within the RUN mode, other than the usual RUN mode, RUN-F, HOLD and DEBUG modes mainly for debugging are also available.



The following explains the operation of each mode, after which the conditions (mode transition conditions) are explained.

HALT: All external outputs are switched OFF, user program execution and I/O processing are halted. In the HALT mode the mode control is run periodically (every 50 ms), idle time is shared to peripheral support and diagnostic control. Externally this is the mode for creating/amending user programs.

RUN: After initial load (where necessary), user data initialization (where necessary), I/O module mounting check, user program check, and scan mode decisions, T3 goes into the RUN mode. Mode control, batch I/O processing timer update, and user program execution are run repeatedly in the RUN mode. This is called scan control. There are 2 scanning methods; the floating scan repeats program execution continuously and the constant scan repeats program execution in a fixed cycle. Selection is called scan mode selection. Scan control is explained in detail in 2.4.

RUN-F: This is the forced run mode. It differs from the above RUN mode in that scan control begins even if the allocated I/O modules are not actually mounted. (If other modules are mounted instead, the mode will not run.)
Otherwise action is the same as the above RUN mode.

HOLD: This is the scan temporary stop mode. Only the batch I/O processing is run, the timer update and the user program execution are halted. The scan mode continues from the status previously reached.
The I/O module test can be performed by the data monitor/set function.

DEBUG: This is the mode which may be used for program debugging functions (single step execution, single rung execution, N scan execution, breakpoint setting, etc).
In this mode, there are three sub-modes; D-HALT, D-STOP and D-RUN.
For the DEBUG mode functions, see Section 5.11.3.

ERROR: When an error is detected in one of the diagnostic checks and operation cannot be resumed by the prescribed retry action, T3 will enter this mode. In the ERROR mode the output is completely OFF, only the error reset command is effective from the programmer (the error reset command takes T3 back to the HALT mode). Refer to 5 RAS Functions for detailed diagnosis.

The transition conditions for each mode are shown below.

- HALT mode transition conditions

Previous state			OP mode transition factor	OP mode after transition	Note
OP mode	RAM/ROM	Mode SW			
(Power off)	RAM	—	Power on	HALT	INZ
	ROM	HALT	Power on		IL, INZ
ERROR	—	—	Command Error Reset		
Other than above	—	RUN	Mode SW → HALT		
		RUN/P-RUN	Command HALT		

- RUN mode transition conditions

Previous state			OP mode transition factor	OP mode after transition	Note
OP mode	RAM/ROM	Mode SW			
(Power off)	ROM	RUN	Power on	RUN	IL, INZ
		P-RUN	Power on		INZ
	—	RUN/ P-RUN	Power on (HOT restart)		
HALT	RAM	HALT	Mode SW → RUN		INZ
		RUN/P-RUN	Command RUN		INZ
	ROM	HALT	Mode SW → RUN		IL, INZ
		RUN	Command RUN		IL, INZ
		P-RUN	Command RUN		INZ
HOLD	—	RUN/P-RUN	Command HOLD Cancel	RUN or RUN-F	Return to mode before HOLD

- RUN-F mode transition conditions

Previous state			OP mode transition factor	OP mode after transition	Note
OP mode	RAM/ROM	Mode SW			
HALT	RAM	RUN/P-RUN	Command Force Run	RUN-F	INZ
	ROM	RUN	Command Force Run		IL, INZ
		P-RUN	Command Force Run		INZ
HOLD	—	RUN/P-RUN	Command HOLD Cancel	RUN-F or RUN	Return to mode before HOLD

- HOLD mode transition conditions

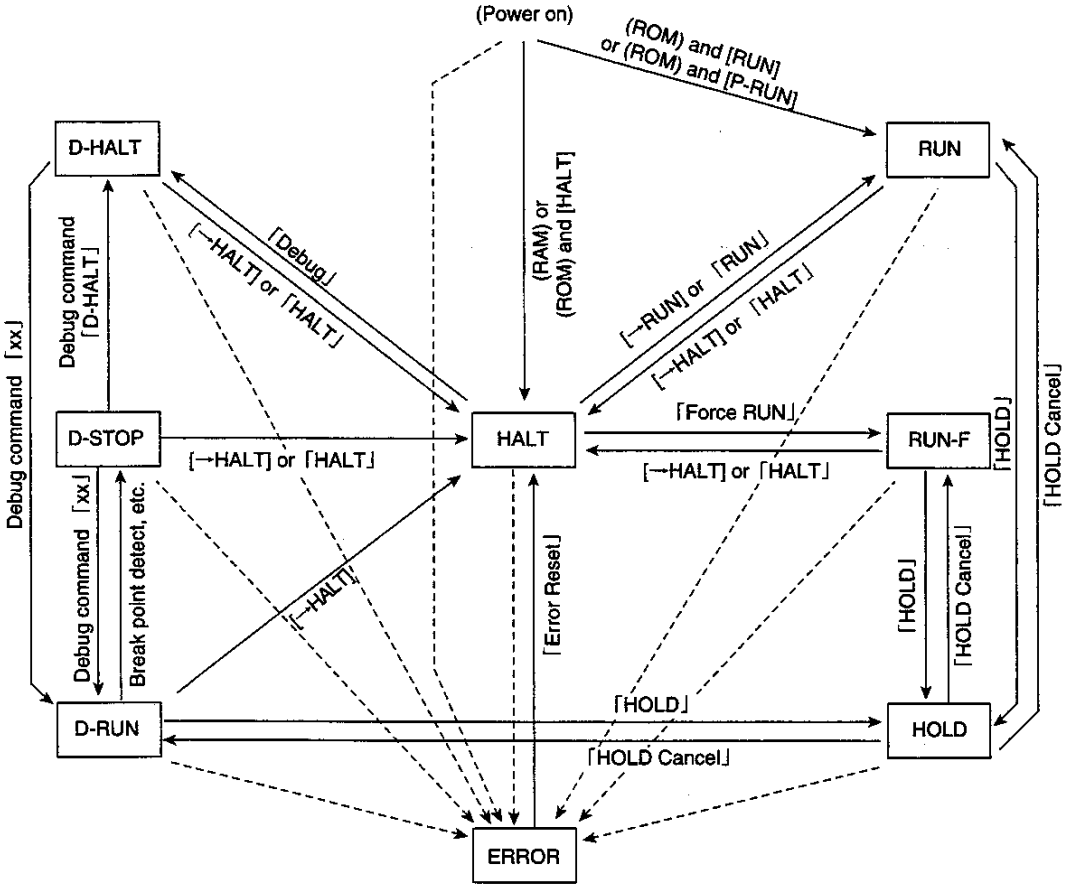
Previous state			OP mode transition factor	OP mode after transition	Note
OP mode	RAM/ROM	Mode SW			
RUN	—	RUN/P-RUN	Command HOLD	HOLD	
RUN-F	—	RUN/P-RUN	Command HOLD		
D-RUN	—	RUN/P-RUN	Command HOLD		

- DEBUG mode transition conditions

Previous state			OP mode transition factor	OP mode after transition	Note
OP mode	RAM/ROM	Mode SW			
HALT	—	RUN/P-RUN	Command Debug	D-HALT	
D-STOP	—	RUN/P-RUN	Command D-HALT		
D-HALT	—	RUN/P-RUN	Command Initial	D-RUN	INZ
			Command Continue		
			Command Step		
			Command Rung		
D-STOP	—	RUN/P-RUN	Command Initial		INZ
			Command Continue		
			Command Step		
			Command Rung		
HOLD	—	RUN/P-RUN	Command HOLD Cancel		
D-RUN	—	RUN/P-RUN	N scan completed	D-STOP	
			Break point detected		
			Stop condition fulfilled		
			Step exeution completed		
			Rung execution completed		
			Command Stop		
D-HALT	—	RUN	Mode SW → HALT	HALT	
		RUN/P-RUN	Command HALT		
D-STOP	—	RUN	Mode SW → HALT		
		RUN/P-RUN	Command HALT		
D-RUN	—	RUN	Mode SW → HALT		

*1) In the table, OP mode, RAM/ROM and Mode SW mean Operation mode, RAM/ROM switch and Operation mode switch, respectively.
*2) — means the switch status is not related to.
*3) In the OP mode transition factor column, "Mode SW→XX" means switching the Operation mode switch to XX position. And "Command XX" means issue of the command XX from the programmer.
*4) Switching the Operation mode switch between RUN and P-RUN will not affect the operation mode. However, the protect state will be changed accordingly. (Refer to Section 5.4).
*5) In the Note column, IL means initial load execution, and INZ means the user data initialization.
*6) See Section 5.11.3 for the DEBUG mode functions.

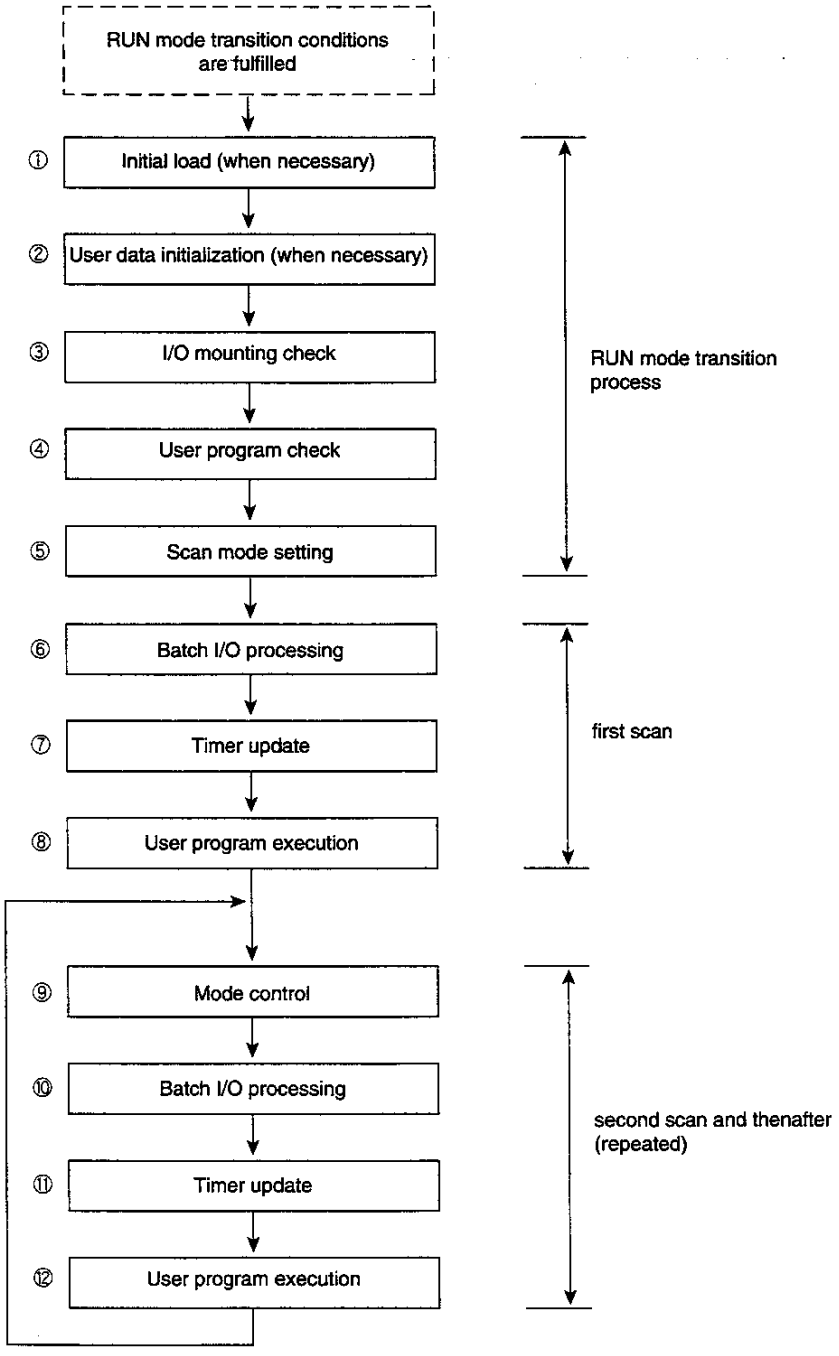
The following diagram illustrates the mode transition conditions.



*1) --- means the ERROR mode transition.
*2) [→XX] means switching the Operation mode switch to the XX position.
*3) 「 XX 」 means issuing of the command XX from the programmer.
*4) The setting status of the RAM/ROM switch and the Operation mode switch at power on are indicated by (XX) and [XX], respectively.

2.4 Scan control

As explained in 2.3, when the RUN mode transition conditions are fulfilled, initial load (when necessary), user data initialization (when necessary), I/O mounting check, program check and scan mode setting are performed, and scan control begins. In scan control, mode control, batch I/O processing, timer update and user program execution are repeated. The following diagram shows the scan control flow chart.


RUN mode transition conditions are fulfilled
① Initial load (when necessary)
② User data initialization (when necessary)
③ I/O mounting check
④ User program check
⑤ Scan mode setting
⑥ Batch I/O processing
⑦ Timer update
⑧ User program execution
⑨ Mode control
⑩ Batch I/O processing
⑪ Timer update
⑫ User program execution
RUN mode transition process
first scan
second scan and thenafter (repeated)

① Initial load
When the RAM/ROM switch is in the ROM side and the Operation mode switch is in the RUN position, the user program and the leading 4k words of the data register (D0000 to D4095) stored in the peripheral memory (IC memory card or the EEPROM) will be transferred to the main memory (RAM) in accordance with the following conditions.

- The initial load will not be performed if both an IC memory card and the EEPROM are not present.
- When an IC memory card and the EEPROM are both present the IC memory card will turn to the transfer source.
- The EEPROM will be the transfer source if no user program is written on the IC memory card installed.
- Initial load will not be performed if the user program is written in the IC memory card or the EEPROM but the contents are destroyed (BCC error detection). In this case, the T3 will enter the ERROR mode.
- Initial load will not be performed if the T3 is in the Hot restart mode from power interruption.

② User data initialization
User data initialization takes place. Refer to 2.2, System initialization, for detailed initialization. User data initialization will not be performed if the T3 is in the Hot restart mode from power interruption.

③ I/O mounting check
The I/O module mounting status is checked based on the I/O allocation information. (Refer to details in 5 RAS functions)

④ User program check
BCC check will be performed on the user program in the main memory (RAM). (Refer to 5 RAS functions for details)

⑤ Scan mode setting
Setting of the scan mode (floating scan or constant scan) will be performed. The scan mode is explained in 2.4.1.

⑥, ⑩Batch I/O processing
Data exchange between the I/O image table (I/O register/device) and the I/O module will be performed based on the I/O allocation information. Data exchange with the data transmission module (TOSLINE-S20, TOSLINE-F10) will be also performed. The first scan is input only.
Batch I/O processing is explained in 2.4.2.

⑦, ⑪Timer update
The activated timer registers and the timing relays (S0040-S0047) will be updated. Timer update is explained in 2.4.3.

⑧, ⑫User program execution
User program instructions will be executed in sequence from the beginning to the END instruction. The execution object is a main program and sub-programs.
In case of an interrupt program, when the interrupt is generated, the corresponding interrupt program is activated immediately.
The user program execution control is explained in detail in section 3.

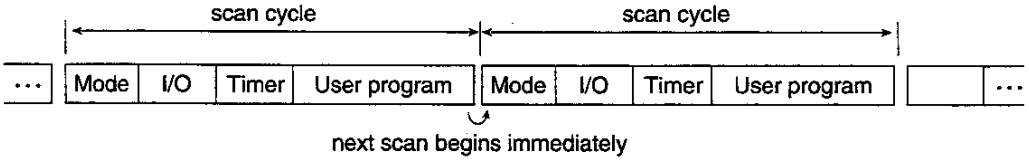
⑨ Mode control
Will check the Operation mode switch and for mode change commands from the programmer and change the operation mode. Also, scan timing control will be performed by measuring the scan cycle.

2.4.1 Scan mode

In the T3, the scan mode enables select from floating scan and constant scan.

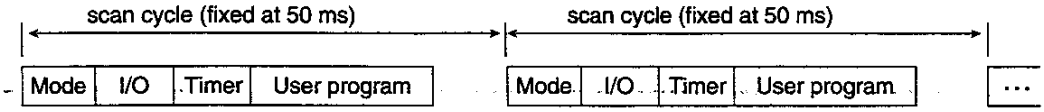
The floating scan mode is that, immediately after one scan is complete the next scan commences. It is the shortest scan cycle but the scan cycle varies according to the user program execution state.

The action of the floating scan is shown in the following diagram.



The constant scan mode has a specified time cycle for scanning. The setup range of the cycle is 10-200 ms (10 ms units). Use this scan cycle to avoid variation in scan intervals.

The action of the constant scan when the cycle is fixed at 50ms is shown in the following diagram.



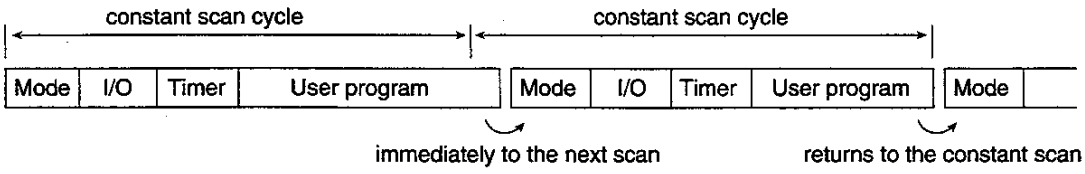
Scan mode selection will be performed by setting up the scan cycle in the system information menu of the programmer.

To select floating scan, do not set up a scan time (leave blank).

With the constant scan, scan time can be set up within the range 10-200ms (10ms units).

NOTE

In the constant scan, if the time for one scan exceeds a specified cycle, it will turn to floating scan, and the constant scan delay flag (special relay-S0008) comes ON. Also, when the scan time reverts to within the specified cycle, the scan cycle will return to the original constant scan.



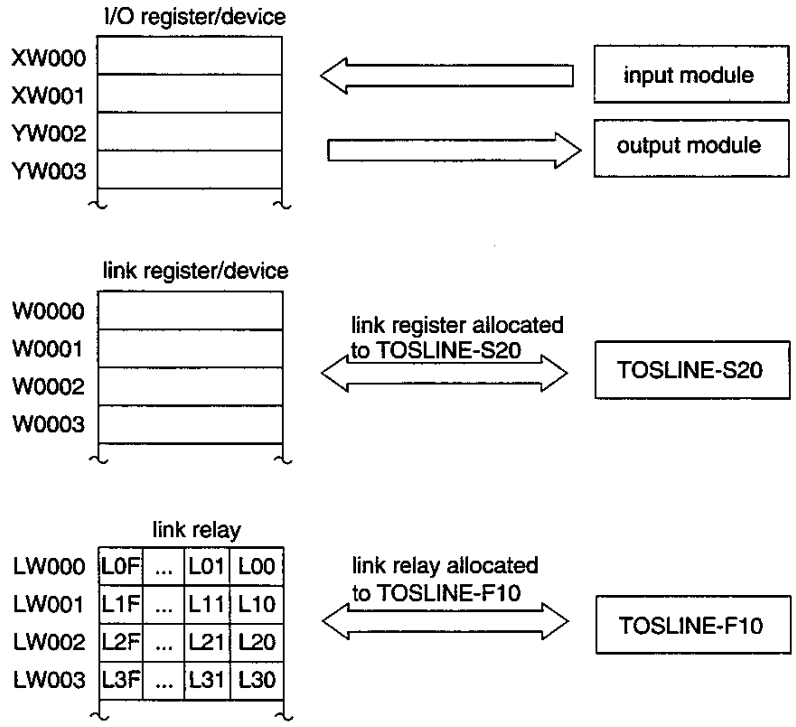
2.4.2 Batch I/O processing

The status of the external input signals will be read from input modules onto the I/O register/device (XW/X). Output register/device (YW/Y) status will be output to the output modules. This process takes place before user program execution and is done in batches, hence named batch I/O processing. The object of the batch I/O processing is as follows:

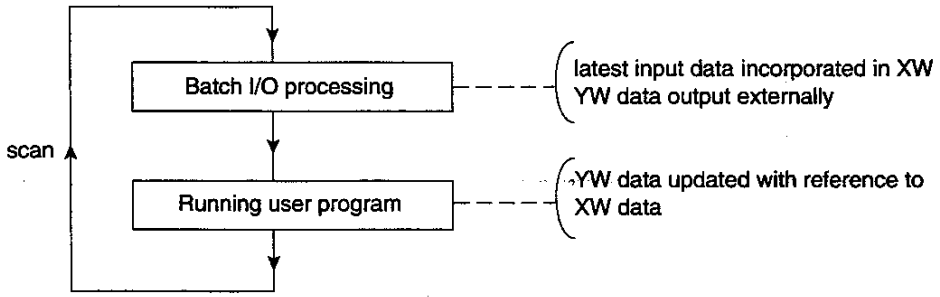
Batch input ... signals from input modules without i designation on I/O allocation and input registers/device (XW/X) which are not forced.

Batch output ... output registers/devices (YW/Y) corresponding to output modules without i designation on I/O allocation

Also, data reading/writing between the data transmission module (TOSLINE-S20,TOSLINE-F10) and the link registers/relays (W/Z and LW/L) will be performed in this process.



If we consider T3 operation simply from the viewpoint of external signal exchanges, batch I/O processing and user program execution can be considered to be repeated continuously, as shown in the following diagram.



Basically, this has the advantage that high speed scanning is achieved because the T3 CPU does not access to the I/O modules during user program execution. Also it is easy to create program logic because the XW data are not changed during user program execution. This method is called the batch I/O processing method (refresh method).

There is also another method of T3 operation whereby I/O module data exchange takes place during user program execution, using IW/I instead of XW/X, and OW/O instead of YW/Y. This method is called the direct I/O processing method. It is recommended that the I/O modules used in direct I/O are inhibited from batch I/O (they have i specification on I/O allocation) to shorten the time for batch I/O processing.

NOTE

(1) Use the following criteria for batch I/O processing time.
Input (XW) ···54 µs/register
Output (YW) ···43 µs/register
Link (W/LW) ···30 µs/register

(2) I/O modules with i designation on I/O allocation (iX, iY, iX+Y) are not part of batch I/O processing. Refer to Part 3 for I/O allocation.

(3) Forced input devices (X), link register devices (Z), and link relays (L) are not part of batch I/O processing. The force function is explained in section 5.11.1.

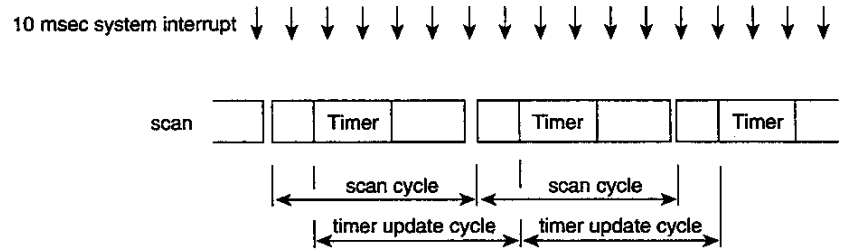
(4) Refer to the data transmission module manual for the allocation of the link register/relay (W/Z and L/LW) to the data transmission module.

(5) With direct I/O processing, output will be in register units even when the bit (O) is specified. Refer to Part 3 for direct I/O registers.

2.4.3 Timer update

The timer registers activated by timer instructions will be updated (increased), and the timing relays (S0040-S0047) will be updated.

- Updating timer registers



The number of system interrupts which occur during the timer update cycle (≈ scan cycle) will be counted, and the counts will be added up in the timer registers which are started up by the timer instructions (TON, TOF, SS, TRG).

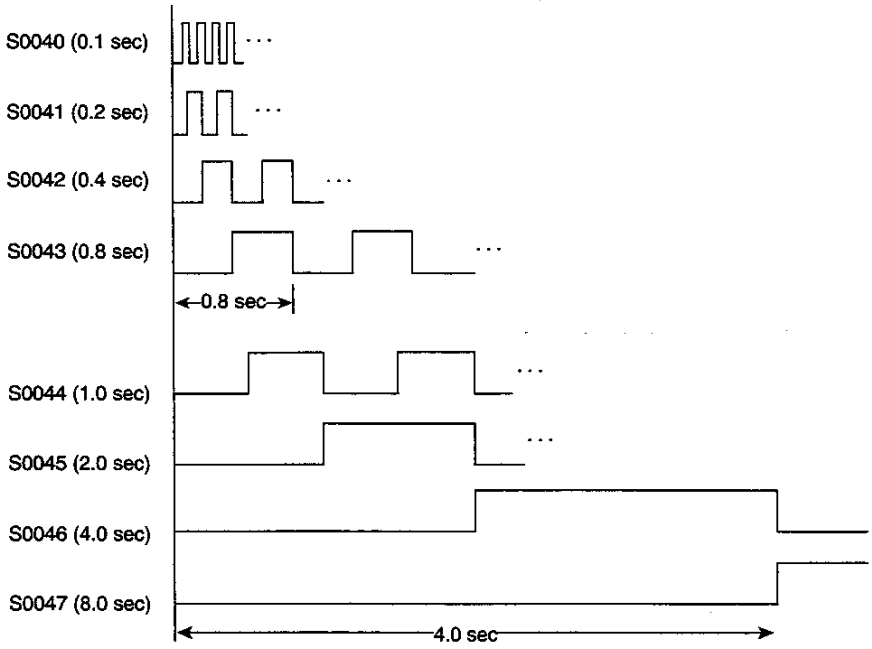
The 10 msec interrupt is used for the 0.01 second timer (T000-T063), the 10 ms interrupts are accumulated and used for the 0.1 second timer (T064-T511). The timer reset and the time-up processing will be performed in the execution of the timer instruction.

Timer classification	Timer register (Timer device)	Preset range	Notes
0.01 second timer	T000-T063 (T.000-T.063)	0-32767 (0~327.67 seconds)	On-delay timer (TON) Off-delay timer (TOF) Single shot timer (SS) Timer trigger (TRG)
0.1 second timer	T064-T511 (T.064-T.511)	0-32767 (0~3276.7 seconds)	

*) Take the criteria for the time for performing the timer register update as follows.

27 μs/timer register (update time)

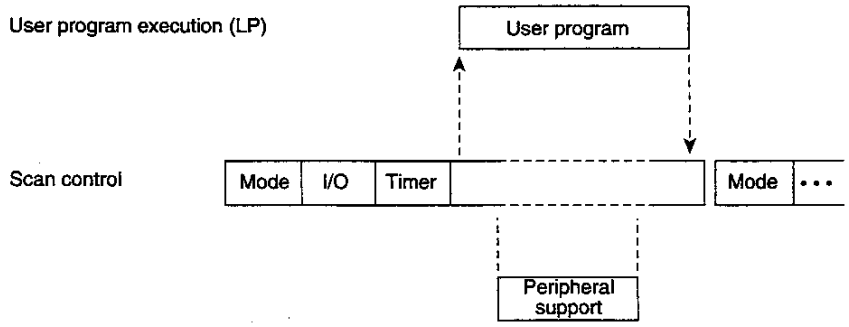
- Updating timing relays
  The timing relays (S0040-S0047) ON/OFF status is controlled by using the 10 msec system interrupt. The binary counter is configured as shown on the next page. (When RUN is started up they will be all OFF)



2.5 Peripheral support

Peripheral support processing will interpret request commands from the peripherals (programmer, computer link, data transmission module), process the requests and responds.

In the T3, the Language processor (LP) takes charge of user program execution. The peripheral support processing will be performed by the main processor during user program execution in parallel.



*1) For commands which require accessing to user data, the command interpretation will be performed in parallel and the data accessing will be performed at the bottom of scan at batch for data synchronization.

*2) If two or more commands are received simultaneously from the request sources, the order of priority will be as follows:
Programmer > Computer link > TOSLINE-S20(CH1) > TOSLINE-S20(CH2)

2.6 Programming support functions

The programming support functions are part of the functions realized as a result of peripheral support processing. Detailed programming support functions are explained in separate manuals for the programmer. The explanation here relates to an overview of the functions and their relation to the T3 operation modes.

(1) Memory clear
When the memory clear command is received, the content of the user program memory (RAM) will be initialized and the content of the user data memory (RAM) will be cleared to 0.

(2) Automatic I/O allocation
When the automatic I/O allocation command is received, the types of I/O modules mounted will be read and the I/O allocation information will be stored on the system information. (System information is in the user program memory.)

(3) Reading the I/O allocation information
The I/O allocation information will be read from the system information, and sent to the peripherals.

(4) Writing I/O allocation information
I/O allocation information received from peripherals is stored on the system information.

(5) Reading the system information
The system information (program ID, retentive memory specification, number of steps used, scan mode specification, other) is read and sent to the peripherals.

(6) Writing system information
The system information (user setup items) received from the peripherals is stored in the system information.

(7) Reading the program
In response to a request from peripherals, a specified range of instructions will be read from the user program memory, and sent to the peripherals.

(8) Writing the program
A specified range of instructions is received from peripherals and written onto the user program memory. After writing, the BCC (check code) correction will be carried out immediately.

(9) On-line program change
Changing the content of the user program memory (adding/changing/inserting/deleting) and the BCC correction will be carried out in the RUN mode. This action is performed after completion of one scan, so the scan cycle is extended while this is being processed.

Changing the program on-line is subject to the following restrictions.

- You may not change the number or running order of instructions which are related to the program execution (see below).

  END, MCS, MCR, JCS, JCR, JUMP, LBL, FOR, NEXT, CALL, SUBR, RET, IRET

- You may not change the SFC structure in the SFC program, but you may change the action corresponding to a step and a transition condition. (Ladder diagram part).

(10) Batch reading of program
The content of the user program memory (including the system information) is read and sent to the peripherals.
It is used for the program uploading (T3 → Programmer → Disk).

(11) Batch writing the program
The user program (including the system information) is received from peripherals and will be stored in the user program memory.
It is used for program download (Disk → Programmer → T3).

(12) Search
The instruction/operand specified by peripherals will be searched through the user program memory and their address will be sent to peripherals.

(13) Program check
When the program check command is received, the user program syntax will be checked. The result of this check will be sent to peripherals.

(14) Reading data
The specified data will be read from the user data memory in response to a request from the peripherals, and the data will be sent to the peripherals.

(15) Writing data
User data address and data content received from peripherals will be stored in the user data memory.

(16) Program reading from the IC memory card/EEPROM
Whether the IC memory card is mounted or not and the EEPROM is fitted or not will be checked. The checked IC memory card or the EEPROM content will be transferred to the user program memory and user data memory (RW, T, C, D) of the main memory (RAM). If the IC memory card and EEPROM are both present, the IC memory card is used.

(17) Program writing to IC memory card/EEPROM
Whether the IC memory card is mounted or not and the EEPROM is fitted or not will be checked. If either exists, the content of the user program memory and the user data memory (RW, T, C, D) will be transferred to the IC memory card or EEPROM. If the IC memory card and EEPROM are both present, the IC memory card will be used.

The execution conditions for these functions are shown in the following table.

Function	Execution conditions
Reading I/O allocation information	Always possible
Reading system information	
Reading the program	
Reading data	
Batch reading the program	Possible except in ERROR mode
Search	
Program check	Possible in HALT mode
Program writing to IC memory card/EEPROM	
Memory clear	Possible in the HALT mode and mode switch other than P-RUN
Automatic I/O allocation	
Writing I/O allocation information	
Writing the system information	
Writing the program	
Batch writing the program	
Program reading from IC memory card/EEPROM	
On-line program change	Possible except in the ERROR mode and except in P-RUN
Writing data	Possible except in the ERROR mode, however writing into D0000-D4095 is prohibited in P-RUN

NOTE

If the password function is used, available functions are limited according to the protect level of the password. Refer to 5.13 for the password function.

3.1 Program types

The T3 can run several different program types in parallel (this function is called the multitask function). This function can be used to realize the optimal response time for each application.

The programs are classified into the 3 types below. There are a total of 14 programs.

- Main program (one)
  This program will be executed every scan and forms the main part of the scan.

- Sub-programs (4)
  This program can be activated by other programs. A total of 4 (#1-#4) are provided. (#1 is fixed function)
  In the floating scan, the sub-program will be executed after the main program execution with time limit (user setting). And in the constant scan, the sub-program will be executed in idle time from completion of the main program execution to the beginning of the next scan.
  By means of sub-programs, the main program can be used as fast scanning task, and the sub-programs as slow scanning (background) tasks.

- Interrupt programs (9)
  When the interrupt condition is fulfilled, the T3 will stop other operations and execute the corresponding interrupt program immediately. A total of 5 are provided: one program which starts up at specified intervals (Timer interrupt program), and 8 programs which start up according to interrupt signals from I/O modules with an interrupt function (I/O interrupt programs #1-#8).

  By means of timer interrupt, time critical control can be achieved, and by means of I/O interrupts, I/O responses can take place without affecting the scan cycle.

The sub-programs and the interrupt programs execution method and the execution conditions are explained in this section.

3.2 Main/sub programs execution control

Four sub-programs (Sub#1 to Sub#4) can be registered. They will be executed according to the conditions described in the table below. Sub#1 will be executed only once before the main program execution in the first scan. The function of Sub#2 can be selected from the normal mode or special mode. Sub#3 and Sub#4 are fixed in normal mode function.
In the normal mode, the execution mode can be selected from one time execution or cyclic execution.

No.	Normal/special	One time/cyclic	Operation
Sub#1	N/A	N/A	Executed only once before main program in the first scan. (after I/O processing)
Sub#2	Normal mode when S0403=0	One time mode when S0405=0	Executed when S0409=1. S0409 is reset automatically.
		Cyclic mode when S0405=1	Executed once every specified scans (SW042) during S0409=1.
	Special mode when S0403=1	N/A	Executed only once before main program in the first scan, instead of Sub#1, if S0400=1 and the last power off period is less than 2s.
Sub#3	Normal mode only	One time mode when S0406=0	Executed when S040A=1. S040A is reset automatically.
		Cyclic mode when S0406=1	Executed once every specified scans (SW043) during S040A=1.
Sub#4	Normal mode only	One time mode when S0407=0	Executed when S040B=1. S040B is reset automatically.
		Cyclic mode when S0407=1	Executed once every specified scans (SW044) during S040B=1.

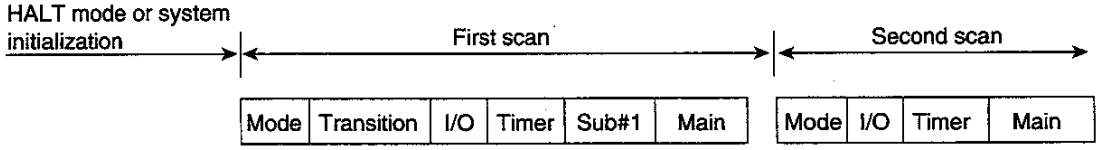
*) Hereafter, the main program, and sub-program #1 to sub-program #4 are referrered as Main, Sub#1 to Sub#4, respectively.

The flags (special relays/registers) related to the sub-program operation are summarized in the table below.

Sub No.	Flag (Name)	Function	Note
Sub#1	S0410 (Sub#1 executing)	0: Not executing 1: Executing	Status
Sub#2	S0400 (Hot restart mode)	0: Normal 1: Hot restart	Setting
	S0403 (Special mode)	0: Normal 1: Special	Setting
	S0405 (Sub#2 mode)	0: One time 1: Cyclic	Setting
	S0409 (Sub#2 start)	0: No request 1: Start request	Command
	SW042 (Sub#2 interval)	Scan number setting for cyclic mode	Setting
	S0411 (Sub#2 executing)	0: Not executing 1: Executing	Status
	S0415 (Sub#2 delay)	0: Normal 1: Delay	Status
Sub#3	S0406 (Sub#3 mode)	0: One time 1: Cyclic	Setting
	S040A (Sub#3 start)	0: No request 1: Start request	Command
	SW043 (Sub#3 interval)	Scan number setting for cyclic mode	Setting
	S0412 (Sub#3 executing)	0: Not executing 1: Executing	Status
	S0416 (Sub#3 delay)	0: Normal 1: Delay	Status
Sub#4	S0407 (Sub#4 mode)	0: One time 1: Cyclic	Setting
	S040B (Sub#4 start)	0: No request 1: Start request	Command
	SW044 (Sub#4 interval)	Scan number setting for cyclic mode	Setting
	S0413 (Sub#4 executing)	0: Not executing 1: Executing	Status
	S0417 (Sub#4 delay)	0: Normal 1: Delay	Status

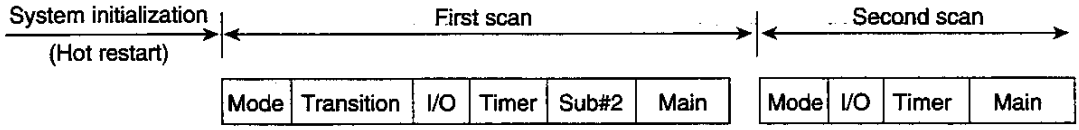
*) In the above table, "Setting" means the user preset flag for execution mode selection, "Command" means the user control flag for activating the sub-program, and "Status" means the execution status flag which can be monitored in the user program.

Sub#1 operation Sub#1 will be executed only once in the first scan before Main execution. Therefore, Sub#1 can be used as the initial setting program at the start of the operation.



Sub#2 special mode operation

If Sub#2 is set as the special mode (S0403=1) and the Hot restart condition is fulfilled (S0400=1 and recovery from power off less than 2 sec), Sub#2 will be executed once in the first scan before Main execution. In this case, Sub#1 is not executed. Also, when the Hot restart condition is fulfilled, the initial load and the user data initialization will not be performed.
Sub#2 special mode can be used as the initial setting program for the restart from power interruption.



Normal mode operation (Sub#2, Sub#3, Sub#4)

In the normal mode, the sub-programs will be executed after the main program execution with time limit. The time assigned for the sub-program execution is different between in the floating scan mode and in the constant scan mode.

In the floating scan mode:
The user sets the sub-program execution time in the system information. The setting range is 1 to 100 ms (1 ms units). The activated sub-program(s) will be executed within this time limit. If the execution cannot finish within this time limit, the execution will be interrupted and re-started in the next scan.

In the constant scan mode:
The activated sub-program(s) will be executed in idle time from completion of the main program execution to the beginning of the next scan. If the sub-program execution cannot finish within this time limit, the execution will be interrupted and re-started in the next scan.

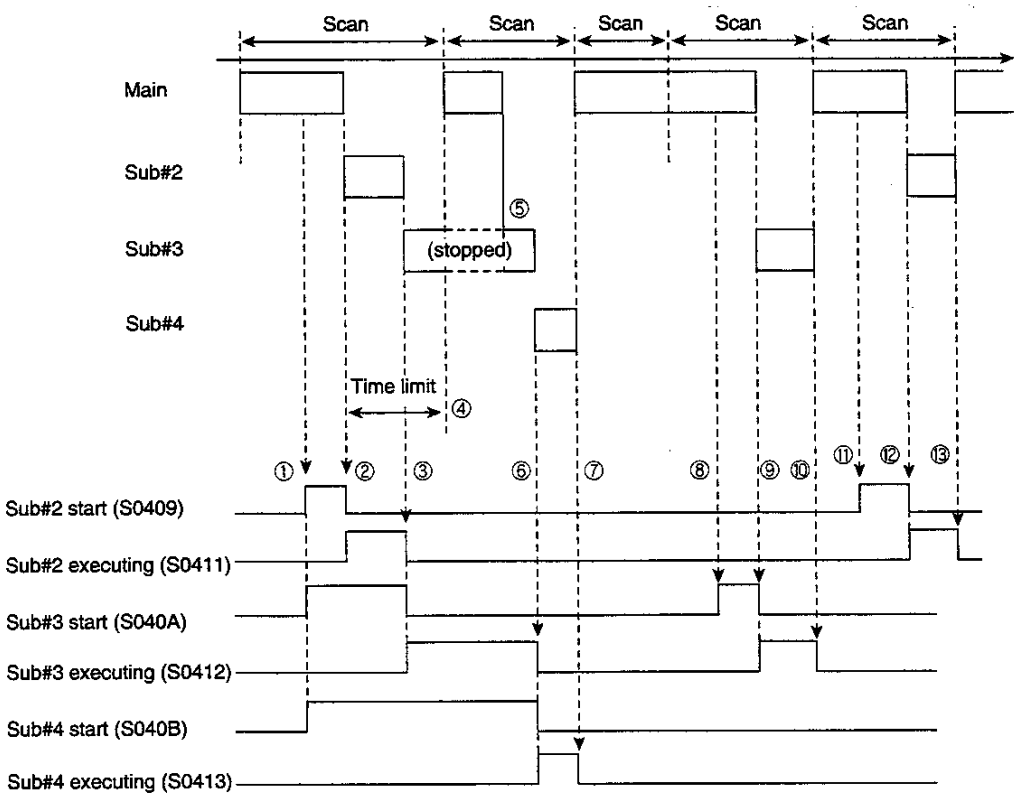
There are two execution modes in the normal mode operation; the one time execution and the cyclic execution.
In the one time mode, the sub-program will be activated when the Sub#n start flag changes from OFF to ON.
In the cyclic mode, the sub-program will be cyclically activated every designated number of scans during the Sub#n start flag is ON.

One time mode The sub-program start request is checked at each time of the main program and the sub-program execution completed. If two or more start requests occur at a time, the order of priority will be as follows.

Sub#2>Sub#3>Sub#4

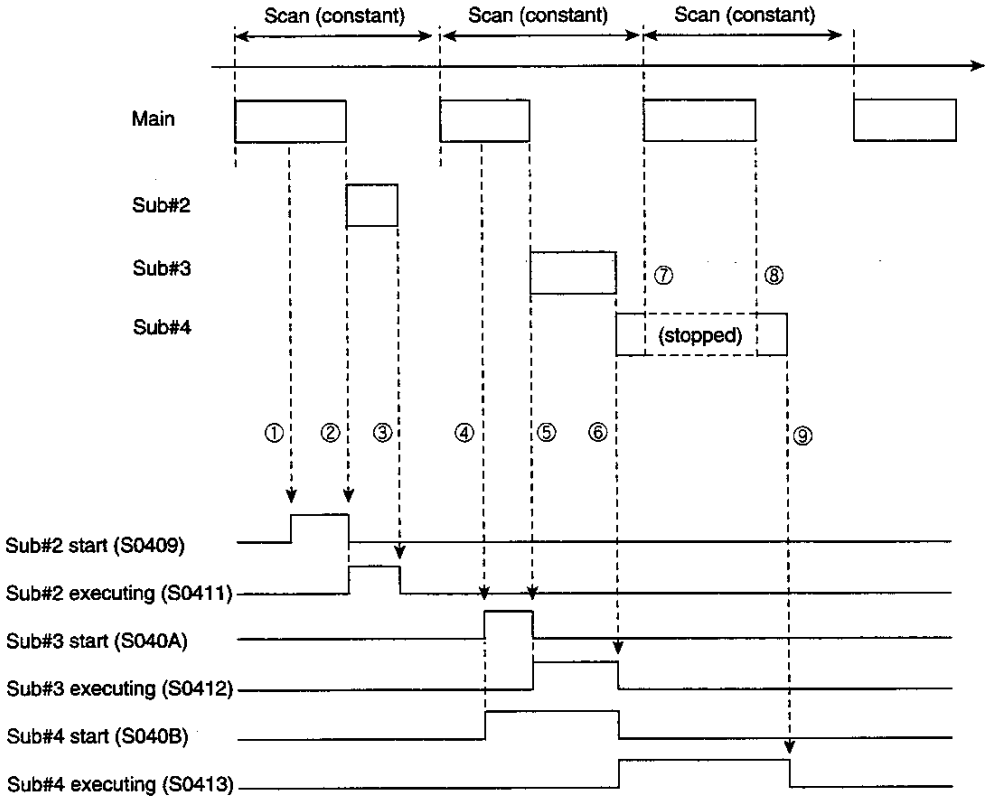
When the sub-program is activated, the start flag is reset automatically.

- Operation example in the floating scan



① Start requests to Sub#2, Sub#3 and Sub#4 from Main
② Sub#2 activated
③ Sub#2 completed and Sub#3 activated
④ Sub#3 interrupted and next scan started
⑤ Main completed and Sub#3 re-started
⑥ Sub#3 completed and Sub#4 activated
⑦ Sub#4 completed and next scan started
⑧ Start request to Sub#3 from Main
⑨ Sub#3 activated
⑩ Sub#3 completed and next scan started
⑪ Start request to Sub#2 from Main
⑫ Sub#2 activated
⑬ Sub#2 completed and next scan started

- Operation example in the constant scan



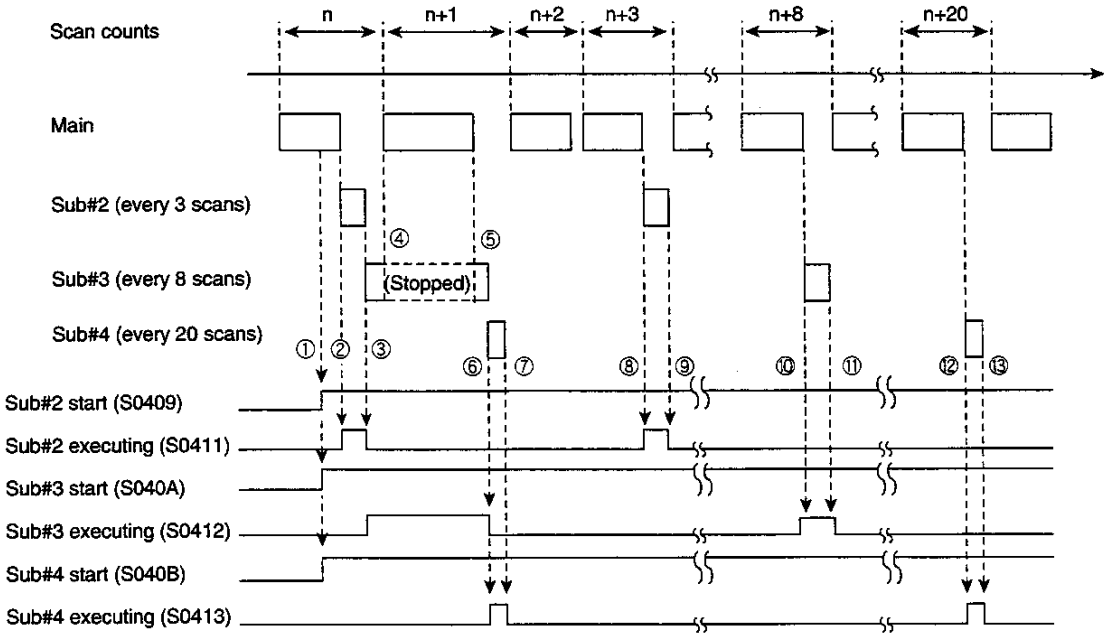
① Start request to Sub#2 from Main
② Sub#2 activated
③ Sub#2 completed
④ Start requests to Sub#3 and Sub#4 from Main
⑤ Sub#3 activated
⑥ Sub#3 completed and Sub#4 activated
⑦ Sub#4 interrupted and next scan started
⑧ Sub#4 re-started
⑨ Sub#4 completed

Cyclic mode While the start flag is ON, the sub-program will be executed once every designated number of scans. The order of execution priority is as follows:

Sub#2>Sub#3>Sub#4

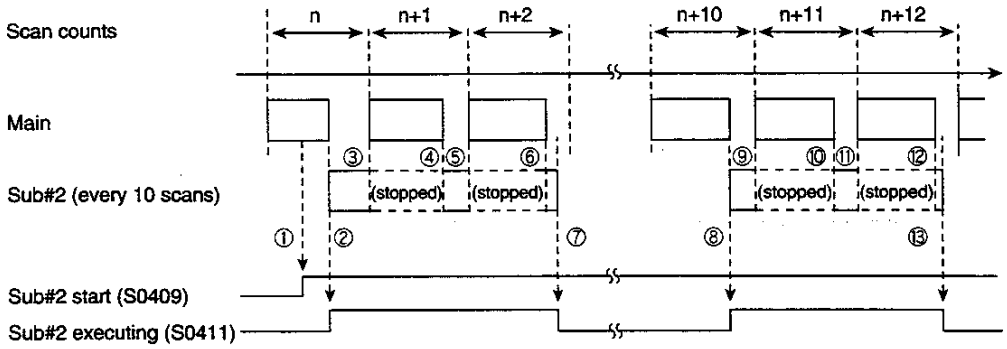
The start flag should be controlled (ON/OFF) by the user program. If the sub-program execution cannot be completed within the designated scans, the delay flag (S0415, S0416, S0417) is set to ON.

- Operation example in the floating scan



① Start requests to Sub#2, Sub#3 and Sub#4 from Main
② Sub#2 activated
③ Sub#2 completed and Sub#3 activated
④ Sub#3 interrupted and next scan started
⑤ Sub#3 re-started
⑥ Sub#3 completed and Sub#4 activated
⑦ Sub#4 completed
⑧ Sub#2 activated in the first scan of next 3 scans
⑨ Sub#2 completed
⑩ Sub#3 activated in the first scan of next 8 scans
⑪ Sub#3 completed
⑫ Sub#4 activated in the first scan of next 20 scans
⑬ Sub#4 completed

- Operation example in the constant scan (Sub#3 and Sub#4 are omitted)

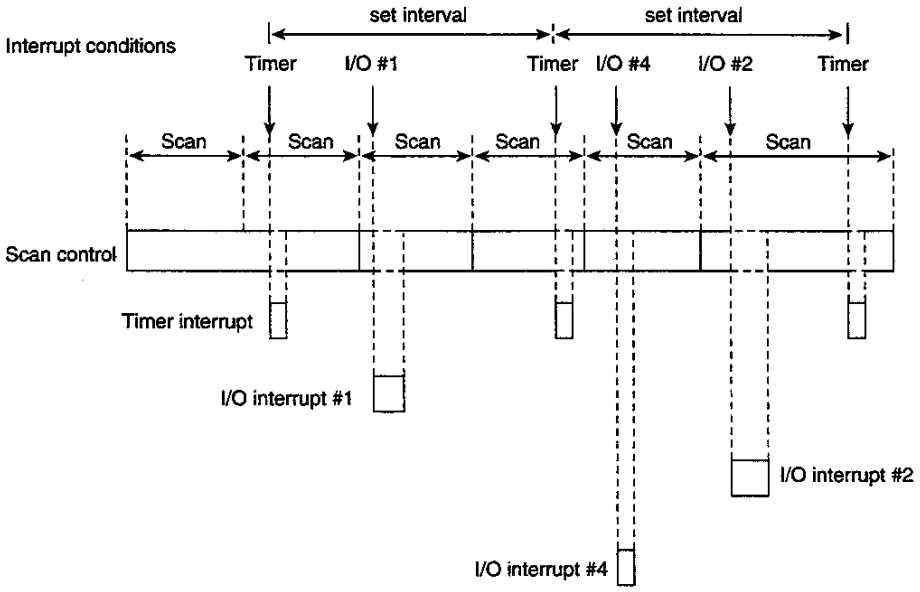


① Start request to Sub#2 from Main
② Sub#2 activated
③ Sub#2 interrupted
④ Sub#2 re-started
⑤ Sub#2 interrupted
⑥ Sub#2 re-started
⑦ Sub#2 completed
⑧ Sub#2 activated in the first scan of the next 10 scans
⑨ Sub#2 interrupted
⑩ Sub#2 re-started
⑪ Sub#2 interrupted
⑫ Sub#2 re-started
⑬ Sub#2 completed

3.3 Interrupt programs execution control

When the interrupt condition is fulfilled, the T3 will stop other operations and execute the corresponding interrupt program immediately. As shown below, you can register one timer interrupt program which starts up according to an interval setup in system information and 8 I/O interrupt programs which start up according to interrupt signals from I/O modules with an interrupt function.

Interrupt program	Operation
Timer interrupt	Activated according to the interrupt interval setup in system information. The interrupt interval is set at 2 to 1000 ms (1 ms units)
I/O interrupt #1 -I/O interrupt #8	I/O interrupt programs are activated by interrupt signals generated from I/O modules with interrupt function



(1) Interrupt priority

When several interrupt conditions occur simultaneously, the programs will be executed in the order of priority shown in the following table (the lower the numerical value the higher the level of priority). Also, if other interrupt conditions occur during an interrupt program execution the interrupt conditions will be put on hold, and after the interrupt program execution is completed, they will be executed in priority order.

Interrupt program	Priority level	Priority in class
Timer interrupt	0	—
I/O interrupt #1	1	0 (initial value)
I/O interrupt #2		1 (ditto)
I/O interrupt #3		2 (ditto)
I/O interrupt #4		3 (ditto)
I/O interrupt #5		4 (ditto)
I/O interrupt #6		5 (ditto)
I/O interrupt #7		6 (ditto)
I/O interrupt #8		7 (ditto)

The timer interrupt has the highest level of priority, followed by the I/O interrupt programs in order.
With respect to the level of priority for I/O interrupt, the I/O interrupt from the module nearest the CPU has the highest level of priority.
Refer to (3) below regarding the correspondence between interrupt programs and I/O modules.

(2) Interrupt enable/disable
You can switch between interrupt disable and enable by using the DI instruction (interrupt disable) and EI instruction (interrupt enable). By executing the DI instruction, the interrupt conditions which occur during interrupt disable mode will be put on hold; these will be then executed instantly when the interrupt enable mode is entered by executing the EI instruction. (DI and EI should be used in a pair) Also, in transition to RUN mode, the interrupt will be disabled in the first scan. It will be enabled automatically from the second scan.

(3) Allocation of I/O interrupt program
The I/O interrupt with the lowest number corresponds to the I/O module with interrupt function nearest the CPU, in the initial state. This allocation can be changed. See Part 3 Section 2.3.3.
There are no restrictions on the mounting position of I/O modules with the interrupt function.

NOTE

The I/O interrupt response time (from the time interrupt conditions arise until interrupt program starts up), with normal interrupt enable and no other interrupt program started up, is an instruction execution time +500 μs in worst case.

4.1 EEPROM support

When the built-in EEPROM type CPU module (PU325) is used, the contents of the user program and the register data can be stored in the EEPROM. They can be read into the main memory (RAM) by the initial load function or programmer operation. Also, the data registers (D) stored in the EEPROM can be accessed from the user program. EEPROM makes it possible to run without battery, and recovery is easy in the event of a program being destroyed.

The following functions are available with EEPROM.

Function	Details	Conditions
Program write into EEPROM	Writes the contents of the user program (including the system information) and the data registers (D), the timer registers (T), the conuter registers (C) and the auxiliary relay registers (RW) in the main memory (RAM) into the EEPROM.	Performed by the 'Program write (RAM→IC card/ EEPROM)' command from the programmer in the following state. · HALT mode · IC memory card is not mounted
Program read from EEPROM	Transfers the contents of the EEPROM to the user program memory, the data registers (D), the timer registers (T), the counter registers (C), and the auxiliary relay registers (RW) in the main memory (RAM).	Performed by the 'Program read (RAM ←IC card/ EEPROM)' command from the programmer in the following state. · HALT mode · IC memory card is not mounted · Mode switch is other than P-RUN
Initial load	Transfers the contents of the EEPROM to the user program memory and the leading 4 k words of the data registers (D0000 to D4095) in the main memory (RAM).	At system initialization: · IC memory card is not mounted · RAM/ROM switch is in ROM · Mode switch is in other than P-RUN
		At transition to RUN mode: · IC memory card is not mounted · RAM/ROM switch is in ROM · Mode switch is in RUN
Read/write the data registers in EEPROM	Reads the data of data registers in EEPROM and stores in the main memory by user program. Writes the specified data of the main memory into the data registers in EEPROM by user program.	Accessed by Expanded data transfer instruction (XFER)

NOTE

(1) Refer to 2.2, System Initialization and 2.4, Scan Control, with respect to the initial load function.
(2) The number of times the EEPROM can be written will be limited by the hardware to 100,000 times. The T3 counts the number of times the EEPROM write is performed. If the 100,000 times is exceeded, the EEPROM alarm flag (S0007) will come ON. However, this checking is not effective for data writing by XFER instruction. It is recommended to check it by user program for the XFER instruction.

4.2 IC memory card support

When an IC memory card is installed in the CPU module, the user program and data can be loaded/saved, in the same way as the EEPROM support function. It facilitates program batch changes and copying etc.

Also, an IC memory card can be used as user data expansion area (expanded file register).

The following functions are available with the IC memory card.

Use type	Function	Details	Conditions
Program auxiliary memory	Program write into IC memory card	Writes the contents of the user program (including the system information) and the data registers (D), the timer registers (T), the counter registers (C) and the auxiliary relay registers (RW) in the main memory (RAM) into the IC memory card.	Performed by 'Program write (RAM→IC card/ EEPROM) ' command from the programmer in HALT mode.
	Program read from IC memory card	Transfers the contents of the IC memory card to the user program memory, the data registers (D), the timer registers (T), the counter registers (C), and the auxiliary relay registers (RW) in the main memory (RAM).	Performed by 'Program read (RAM ←IC card/ EEPROM) ' command from the programmer in HALT mode and mode switch at other than P-RUN.
	Initial load	Transfers the contents of the IC memory card to the user program memory and the leading 4k words of the data registers (D0000 to D4095) in the main memory (RAM).	At system initialization: · RAM/ROM switch is in ROM · Mode switch is other than P-RUN
			At transition to RUN mode: · RAM/ROM switch is in ROM · Mode switch is in RUN
Expansion memory	Sampling trace buffer	Stores trace data when the sampling trace is executed.	Used with the sampling trace function when the MMR allocation is set in the CPU slot.
	Expanded file register	Reads/writes the data in the IC memory card (120k words) as expanded file registers from the user program.	Accessed by the expanded data transfer instruction (XFER) when the MMR allocation is set in the CPU slot.

*1) Refer to 2.2, System initialization and 2.4, Scan Control, for the initial load function.

*2) Refer to Section 5.9 for the sampling trace function, and Part 3 Section 4.2 for the MMR allocation.

5.1 Overview

The meaning of RAS is Reliability, Availability and Serviceability. The RAS function is the general term used for the functions installed in the T3 which increase the reliability and serviceability of the applied systems and support the operation of the system.

This section explains the self-diagnostic functions, maintenance functions, the debugging functions installed in the T3, and the system diagnostic function which can be used by the T3 user.

5.2 Self-diagnosis

The details of the self-diagnosis which are designed to prevent abnormal operation, the timing of the diagnosis and behavior when malfunctions are detected are shown below.

In building up a system, consider the system operation safety in case of the T3 shutdown (fail safe) and the system operation backup function.

In the following explanation, error registration means the storing of the details of the error and the time when it occurred on the event history table; error down means that all the outputs turn OFF and ERROR mode is entered; alarm means that the error is registered, the special relay is set, and running is continued.

(1) Diagnosis at system initialization (when power supply is turned on)

Items	Diagnostics details	Behavior when error detected
System ROM BCC check	The correctness of the system ROM is checked by BCC.	Error registration takes place, FAULT and I/O LED flash. (Programmer communication impossible)
System RAM check	The system RAM read/write is checked.	Error registration takes place, the FAULT LED flashes. (Programmer communication impossible)
Peripheral LSI check	Peripheral LSI is checked for normal initialization. (Read back check)	Error registration takes place, the FAULT LED flashes, the I/O LED lights up. (Programmer communication impossible)
LP check	LP (language processor) is checked for normal initialization.	Error registration takes place, ERROR mode is entered. (Error reset command invalid)
User program memory check	The correctness of the content of the user program memory is checked by BCC. (Checked after initial load when peripheral memory is present)	Error registration takes place, ERROR mode is entered.
User data memory check	The user data memory read/write is checked.	Error registration takes place, ERROR mode is entered. (Error reset command invalid)

Peripheral memory check	The correctness of the peripheral memory (IC memory card or EEPROM) is checked by BCC.	Error registration takes place. ERROR mode is entered.
RTC LSI check	The validity of the data read from the RTC LSI (date and time) is checked. The data is set in the special register.	Alarm. Until reset, the date and time data (in the special register) are HFF.
Battery check	The voltage of the memory backup battery is checked.	Alarm. If the user program memory BCC is normal, it will start up normally. (However, user data in the retentive memory specification is not guaranteed.)

(2) RUN start-up diagnosis

Items	Diagnostics details	Behavior when error detected
I/O verify check	The I/O allocation information and the I/O modules mounted are verified, to check that they agree.	Error registration, error down. However, when start-up is activated by a command from the programmer, a message will the displayed. It remains in HALT mode and no error registration will take place.
I/O bus check	Checks that I/O bus is normal.	Error registration, error down. However, when start-up is activated by a command from the programmer, a message will be displayed. It remain in HALT mode and no error registration will take place.
Expansion unit power check	Checks that power of expansion units is normal	Error registration, error down. However, when start-up is activated by a command from the programmer, it will remain the in HALT mode and no error registration will take place.
I/O response check	Checks that response when I/O module is accessed is within specified response time limits.	Error registration, error down. However, when start-up is activated by a command from the programmer, a message will be displayed. It remain in HALT mode and no error registration will take place.
Program check	User program syntax is checked.	Error registration, error down. However, when start-up is activated by a command from the programmer a message will be displayed. It remain in HALT mode and no error registration will take place.

(3) Diagnosis during scan

Items	Diagnostics details	Behavior when error detected
I/O bus check	Checks that I/O bus is normal. (at batch I/O processing)	Error registration then error down. (However, if recovered by retries, only registration will take place; no error down.)
Expansion unit power check	Checks that power of expansion units is normal. (at batch I/O processing)	Error registration then error down. (However, if recovered by retries, only registration will take place; no error down.)
I/O response check	Checks that response when I/O module is accessed is within specified response time limits. (At batch I/O processing and at direct I/O instruction)	Error registration then error down. (However, if recovered by retries, only registration will take place; no error down.)
I/O bus parity check	Bus parity is checked when the I/O module is accessed. (At batch I/O processing and direct I/O instruction)	Error registration then error down. (However, if recovered by retries, only registration will take place; no error down.)
LP function check	Test program run in LP (language processor) and checked for correct results. (When running the user program)	Error registration then error down. (However, if recovered by retries, only registration will take place; no error down.)
LP illegal instruction detection check	Checks whether or not illegal instruction is detected in LP (language processor). (When running the user program)	Error registration and then error down.
Scan time over check	Checks that scan cycle does not exceed set value (200ms). However, set value can be chaged by user instruction (WDT). (When running the user program)	Error registration and then error down.

(4) Diagnosis in any mode (executed in background)

Items	Diagnostics details	Behavior when error detected
System ROM BCC check	The correctness of the system ROM is checked by BCC.	Error registration and then error down. (Error reset command invalid)
System RAM check	The system RAM read/write is checked.	Error registration and then error down. (Error reset command invalid)
Peripheral LSI check	Peripheral LSI setting status is checked.	Error registration and then error down. (Error reset command invalid)
Watchdog timer check	Watchdog timer system runaway check. (Set at 350ms)	Error registration and transition to ERROR mode after system reset.
User memory check	User memory (RAM) read/write checked.	Error down after error registration (with retry)
LP check	LP (language processor) read/write is checked.	Error registration and then error down.
Battery check	Memory backup battery voltage checked.	Alarm
RTC LSI check	Date and time data read from RTC LSI every 300ms, validity checked, data set in special register.	Alarm. Until reset, date and time data are HFF.

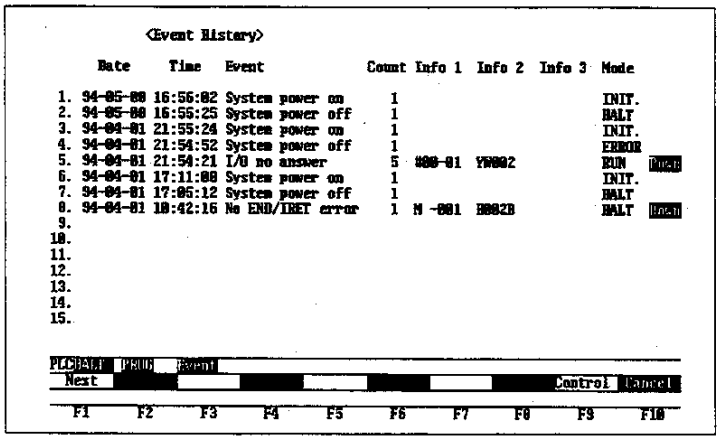
NOTE

Refer to the separate T3 User's Manual-Hardware, for details of troubleshooting.

5.3 Event history

When an error is detected by the T3 diagnosis, the details and time of occurrence will be registered in the event history table (besides errors, the times power ON/OFF are also registered). The 30 most recent occurrences of errors can be registered in the event history table. As new data is registered, the data registered previously will be shifted down in sequence, and the oldest data will be deleted.

Use the event history table for maintenance information. It can be displayed on the programmer as below. The contents of the event history table are remained until executing the event history clear command or the memory clear command from the programmer.


<Event History>
Date Time Event Count Info 1 Info 2 Info 3 Mode
1. 94-05-00 16:56:02 System power on 1 INIT.
2. 94-05-00 16:55:25 System power off 1 HALT
3. 94-04-01 21:55:24 System power on 1 INIT.
4. 94-04-01 21:54:52 System power off 1 ERROR
5. 94-04-01 21:54:21 I/O no answer 5 #00-01 Y0002 RUN
6. 94-04-01 17:11:00 System power on 1 INIT.
7. 94-04-01 17:05:12 System power off 1 HALT
8. 94-04-01 10:42:16 No END/IRET error 1 M -001 H002B HALT
9.
10.
11.
12.
13.
14.
15.
PLC Edit PROG Event
Next Control Cancel
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10

The meaning of each item on the screen above is as follows.

(1) Number (1-30)
Indicates the order of occurrence. Number one is the most recent.

(2) Date (year-month-day)
Indicates the date of occurrence. This is shown as "??-??-??" if the RTC LSI is abnormal.

(3) Time (hours: minutes: seconds)
Indicates the time of occurrence. This is shown as "??:??:??" if the RTC LSI is abnormal.

(4) Event
Indicates the sort of error detected. (System power on and system power off are also registered.)

(5) Count
Indicates the number of times the error was detected. For example, an error is detected during a process, the retry is repeated 4 times, the malfunction does not change and it goes to error down. This is indicated as count 5 and DOWN will be displayed under the Mode.

(6) Information 1, Information 2, Information 3
Indicates supplementary information regarding the error. For example, with an I/O error the I/O module position (unit No, slot No) where the error occurred and the read/write register address etc. will be indicated.

(7) Mode
Indicates the actual mode when the error was detected. Also displays DOWN when error down occurs. On the mode display, INIT indicates the system initialization after power is turned on.

*) Refer to the separate T3 User's Manual-Hardware for display details of detected errors and methods of proceeding.

5.4 Memory protect function

The CPU module operation mode switch has 3 positions, HALT/RUN/P-RUN. The P-RUN position has a memory protect function.

With the operation mode switch in the P-RUN position, the following operations cannot be carried out. The message "Memory protected" will be displayed on the programmer screen if the user tries to do so.

(1) Memory clear
(2) I/O automatic allocation
(3) Write I/O allocation information
(4) Write system information
(5) Program editing (including on-line changes)
(6) Program downloading
(7) Program read from IC card/EEPROM (including initial load)
(8) Write data to the leading 4k words of data register (D0000-D4095)

The memory protect function can prevent the program being destroyed due to incorrect operation of the programmer.

NOTE

When the operation mode switch is in P-RUN position, data writing to the leading 4k words of data register (D0000 to D4095) by the following instructions is disabled.

- Calendar operation (CLDS FUN 155)
- Expanded data transfer (XFER FUN 236)
- Special module data read (READ FUN 237)

Also, D0000 to D4095 are not cleared in the user data initialization.

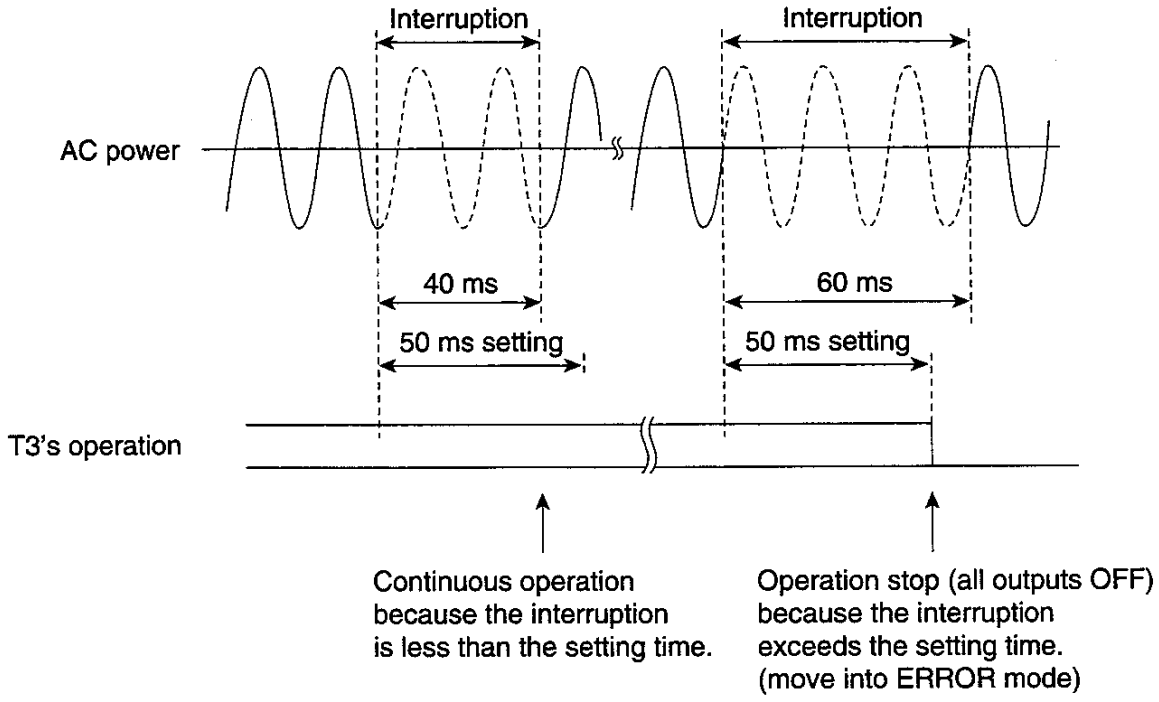
5.5 Power interruption detection function

The T3 has two functions that control the T3's operation in the event of power interruption. One is the power interruption shutdown function which shuts down the T3's operation when power off state continues for designated time. The other is the hot restart function which enables the restart from the power interruption without initialization.

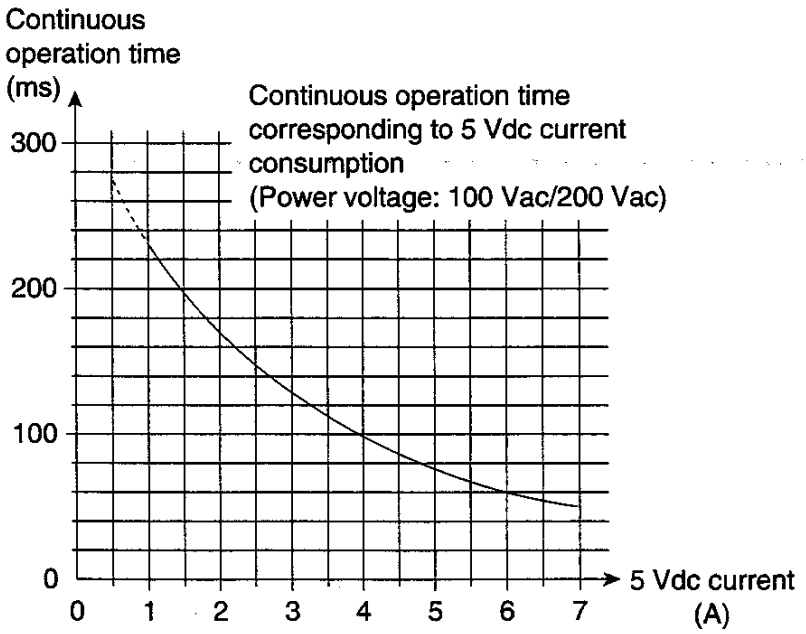
5.5.1 Power interruption shutdown function

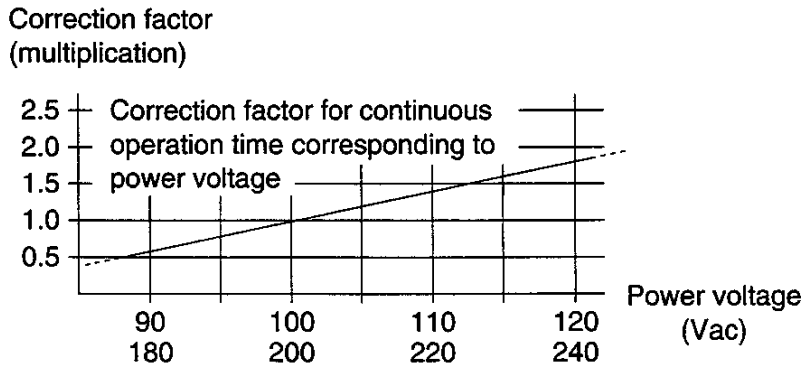
Normally, when power is turned off, the T3 continues operation until the internal 5 Vdc drops to the specified voltage. This continuous operation time varies depending on the load condition of the 5 Vdc. In some application, fixing the continuous operation time will be required.

The T3 has the function to designate the continuous operation time within the power supply module capacity. This function is called the power interruption shutdown function. The continuous operation time can be set by the programmer's System Diagnosis menu or by writing the value into the special register SW045. (10-255 ms, 1 ms units)


Interruption
Interruption
AC power
40 ms
60 ms
50 ms setting
50 ms setting
T3's operation
Continuous operation because the interruption is less than the setting time.
Operation stop (all outputs OFF) because the interruption exceeds the setting time. (move into ERROR mode)

This function is effective when the setting time is within the maximum continuous operation time. The maximum continuous operation time can be evaluated by the internal 5 Vdc current consumption and the power voltage, as the following graphs.





NOTE

(1) DC power supply module (PS332) dose not support this function.
(2) When the power interruption continues for the setting time, the T3 will switch all outputs to OFF and enter the ERROR mode. After that, if power recovers within the maximum continuous operation time, the T3 will not return to RUN mode automatically.
(3) Refer to separate T3 User's Manual-Hardware for internal 5 Vdc current consumption of each module.

5.5.2 Hot restart function

For the T3, the user can decide the operation re-start condition at the recovery from the power interruption.

The hot restart function will be effective when the special relay S0400 is set to ON (S0400=1). In this case, if power is turned off in the RUN mode and recovered within 2 seconds, the T3 moves into RUN mode without the initial load and the user data initialization.

By using this function together with the special mode of the sub-program #2, the user can decide the operation re-start condition as follows:

Interruption time	Re-start condition	Method
Longer than 2 seconds	Re-start after the normal initialization	–
Within 2 seconds	Re-start after the normal initialization	Do not use the hot restart function (S0400=0)
	Re-start after setting the prespecified data into registers/devices	Use sub-program #2 as special mode to set prespecified data
	Re-start after setting the data according to input status	Use sub-program #2 as special mode to set data according to input status
	Re-start without any initialization (hot restart)	Do not use sub-program #2 special mode

NOTE

(1) When power interruption is longer than 2 seconds, normal initialization will be carried out even if S0400 is ON.
(2) The hot restart function is also available by using the programmer's System Diagnosis menu in addition to setting S0400 to ON.

5.6 I/O error mapping function

The T3 has the function that enables user to detect abnormality in I/O modules and the mounting position of the abnormal module. This function is called the I/O error mapping function.

To use this function, set the special relay S0150 to ON. When this function is enabled, the T3 checks the status of each I/O modules. And if abnormality is detected, the I/O alarm flag (S0009) is set to ON and the mounting position is registered in the special registers (SW046-SW049).

The target module and the abnormality for this function are as below.

Type	Description	Abnormality
DO333	16 pts DC output	• Fuse blown
DO334	32 pts DC output	• External 24 Vdc abnormal
AC363	16 pts AC output	• Fuse blown
AC364	32 pts AC output	
RO364	32 pts relay output	• External 24 Vdc abnormal
RO363S	16 pts relay output	
AD368	8 ch analog input	• Module CPU error
DA364	4 ch analog output	• External 24 Vdc abnormal
DA374		• Module CPU error
SN321	TOSLINE-S20	• Module CPU error
SN322		
SN323		
MS311	TOSLINE-F10	• Module CPU error

5.7 Online I/O replacement function

When a failure has occurred in an I/O module, the module can be replaced without stopping the T3 operation. This function is called the online I/O replacement function.

To use this function, set the I/O disconnect designation for the failed module by the programmer. By this operation, the T3 stops accessing to the module. Then replace the module, and release the I/O disconnect designation.
By the above procedure, the online I/O replacement can be performed.

This function is effective for the following failures.
- Input photo-coupler failure
- Output device (transistor, triac or relay) failure
- Built-in fuse blown

The modules available for this function are listed below.

Type	Description
DI334/334H	32 pts DC input
DI335/335H	64 pts DC input
IN354/364	32 pts AC input
DO333	16 pts DC output
DO334	32 pts DC output

Type	Description
DO335	64 pts DC output
AC363	16 pts AC output
AC364	32 pts AC output
RO364	32 pts relay output
RO363S	16 pts relay output

NOTE

(1) When the I/O disconnect designation is set for an output module, the output status is frozen (not cleared to OFF).
(2) Pay special attention to the live parts for safety when using this function.
(3) Replacement is available between the modules of the same module type (X 2W, etc.)

5.8 Execution status monitoring

The following functions are served by the T3 for user to monitor the T3 execution status. (Refer to separate manuals for the programmer for operation of these.)

(1) Execution time measurement function
Measures the following execution times. This data can be monitored on the programmer.

- Scan cycle
  current value, maximum value, minimum value (1 ms units)
- Main program execution time
  current value, maximum value, minimum value (1 ms units)
- Sub-program execution time (Sub#1-#4)
  current value, maximum value, minimum value (1 ms units)
- Timer interrupt execution time
  latest value, maximum value, minimum value (0.1 ms units)
- I/O interrupt execution time (I/O #1-#8)
  latest value, maximum value, minimum value (0.1 ms units)

NOTE

(1) The scan cycle value includes the scan overhead and all interrupts occurring during the scan.

(2) With the main program and the sub-program execution times the interrupt time for any interrupts occurring are excluded.

(2) Online trace function
This function traces the status during program execution and displays on the programmer screen (power flow display, register value display). Since this displays data from the point in time that the instruction is executed rather than at the end of a scan cycle, it is useful for program debugging.

5.9 Sampling trace function

The sampling trace function collects the status of specified registers/devices and stores it into the sampling buffer, according to the specified sampling condition. The collected data can be displayed on the programmer screen in the format of trend graph (for registers) or timing chart (for devices).
The sampling trace function is useful for program debugging and troubleshooting.

Sampling buffer

Optional IC memory card or the file registers (F registers) of the T3 CPU module is used for the sampling buffer.

① To use the IC memory card for sampling buffer:
Insert the IC memory card into the T3 CPU module. And set MMR to the PU slot in the I/O allocation. By this operation, the IC memory card is assigned for the sampling buffer. The sampling buffer size is 8k words (fixed).

② To use the file registers for sampling buffer:
When the MMR setting is not made to the PU slot, the file registers are assigned for the sampling buffer. The sampling buffer size can be selected from 1k to 8k words (1k words unit). The size is registered in the system information. The file registers are assigned for the sampling buffer from the highest address according to the size specified. (Do not use the file registers assigned for the sampling buffer in the user program)

Sampling target

The sampling targets (registers/devices) are selected from the following combinations.

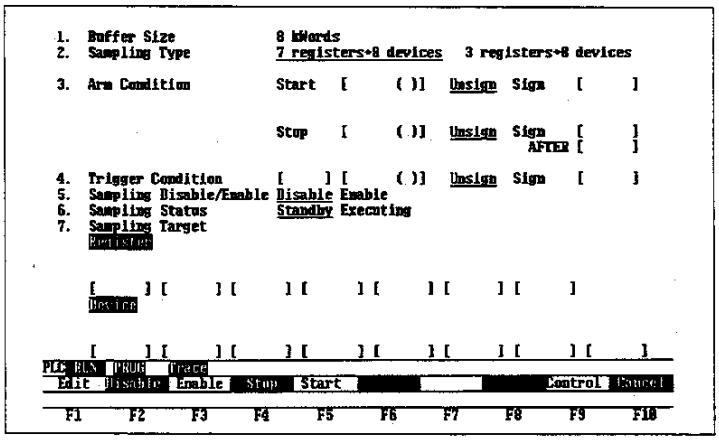
① 3 registers + 8 devices
② 7 registers + 8 devices

In case of ①, 256 times per 1k words (max. 2048 times) of collection is available. In case of ②, 128 times per 1k words (max. 1024 times) of collection is available.

Sampling condition

There are the arm condition and the trigger condition for the sampling trace execution conditions.
The arm condition consists of the start condition and the stop condition. When the start condition is fulfilled, the data collection is started. And when the stop condition is fulfilled, the data collection is stopped. However, if the after counts is added to the stop condition, the arm condition is extended for specified counts of scans after the stop condition is fulfilled.
The trigger condition specifies the timing of the data collection. That is, the data collection is carried out at the moment of the trigger condition is fulfilled while the arm condition is fulfilled.

The sampling target and the condition are set on the programmer screen (below). Setting is available when the T3 is in HALT mode or the sampling trace is disabled by pressing F2 (Disable).


1. Buffer Size 8 kWords
2. Sampling Type 7 registers+8 devices 3 registers+8 devices
3. Arm Condition Start [( )] Unsign Sign [ ]
Stop [( )] Unsign Sign [ ]
AFTER [ ]
4. Trigger Condition [ ] [( )] Unsign Sign [ ]
5. Sampling Disable/Enable Disable Enable
6. Sampling Status Standby Executing
7. Sampling Target
Registers
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
Devices
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
PLC RUN PROG Trace
Edit Disable Enable Stop Start Control Cancel
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10

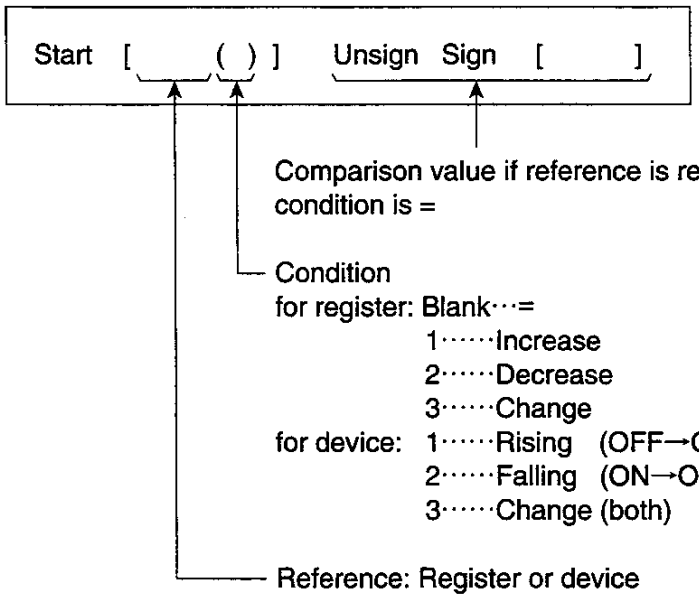
The sampling trace is executed when it is enabled by pressing F3 (Enable).

NOTE

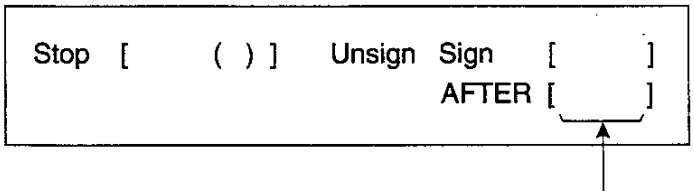
The sampling trace can also be started/stopped by manually without setting the arm condition. F5 (Start) and F4 (Stop) are used.

The setting method for each condition is as follows.

Arm start condition:

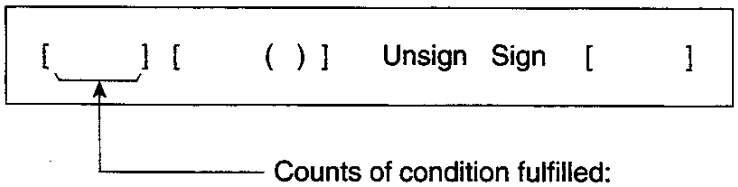


Arm stop condition:



*) Other setting items are the same as the arm start condition.

Trigger condition:

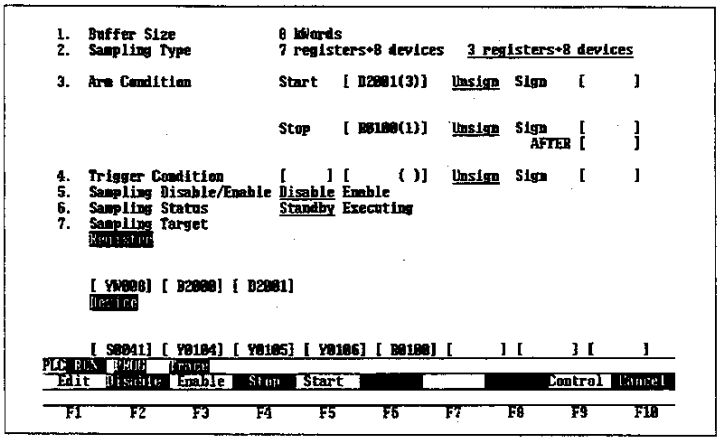

[ ] [( )] Unsign Sign [ ]
Counts of condition fulfilled:
Data collection is carried out once per counts times the following condition fulfilled
Blank ····· No setting (counts = 1)
1 to 65535···Counts

*) Other setting items are the same as the arm start condition

NOTE

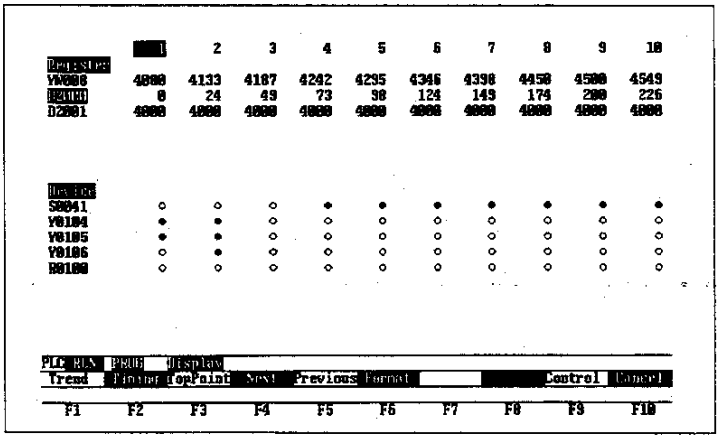
The evaluation of the conditions are performed at the end of every scan.

Execution example Sampling target and condition setting example:

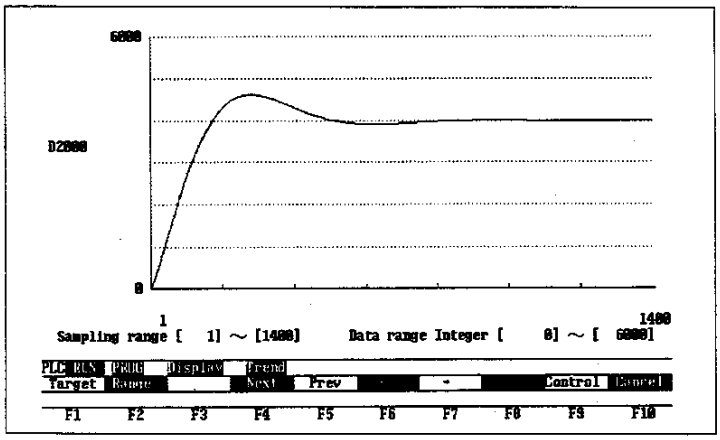

1. Buffer Size 8 kWords
2. Sampling Type 7 registers+8 devices 3 registers+8 devices
3. Arm Condition Start [D2001(3)] Unsign Sign [ ]
Stop [R0100(1)] Unsign Sign [ ]
AFTER [ ]
4. Trigger Condition [ ] [( )] Unsign Sign [ ]
5. Sampling Disable/Enable Disable Enable
6. Sampling Status Standby Executing
7. Sampling Target
[YW008] [D2000] [D2001]
[S0041] [Y0104] [Y0105] [Y0106] [R0100] [ ] [ ] [ ]
PLC RUN RMON Trace
Edit Disable Enable Stop Start Control Cancel
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10

In the above example, the data of YW008, D1000, D2001, S0041, Y0104, Y0105, Y0106 and R0100 are collected every scan, for the duration of from D2001 changed to 10 scans after R0100 changed to ON.

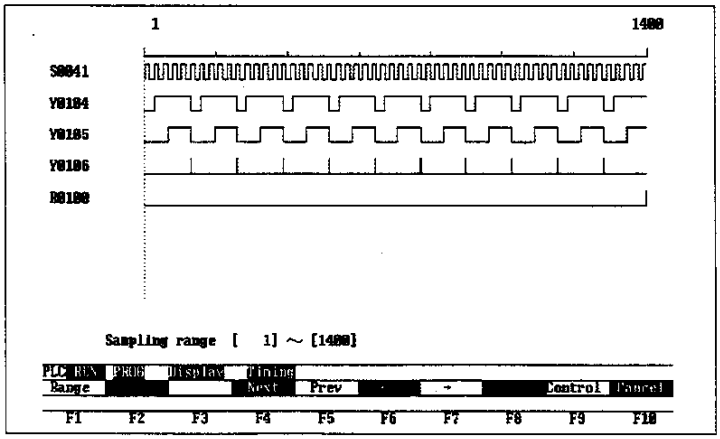
Data display example 1 (Data):



Data display example 2 (Trend graph):


6000
D2000
0
1
1400
Sampling range [1] ~ [1400] Data range Integer [0] ~ [6000]
PLC RUN PROG Display Trend
Target Range Next Prev Control Cancel
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10

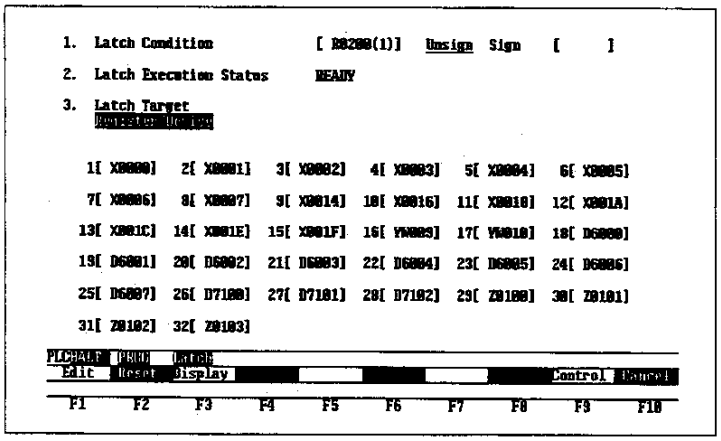
Data display example 3 (Timing chart):



5.10 Status latch function

The status latch function will transfer the specified devices/registers data in batches to the internal latch data storage area when the latch condition set by the programmer is fulfilled or when the Status latch instruction (STLS) is executed.
The latch condition is evaluated and data collected at the end of the scan. However, when the STLS instruction is executed, the data collection is carried out at the time of the instruction is executed.
Latched data can be displayed on the programmer.
The latched status can be reset by the latch reset command of the programmer or by executing the Status latch reset instruction (STLR).

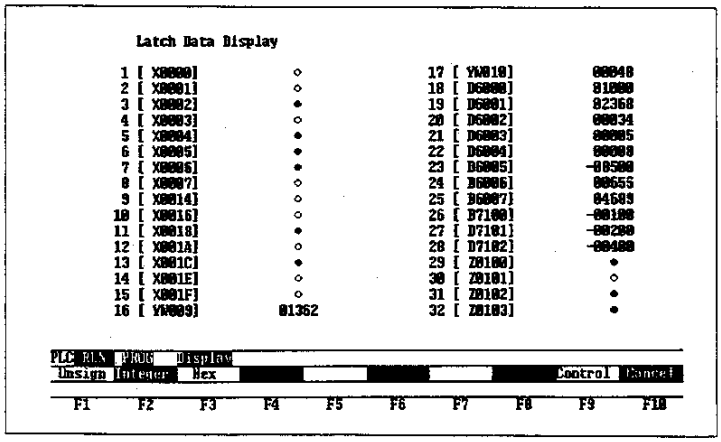
The latch target and condition setting screen is shown below.



The setting method for the latch condition is the same as the arm condition of the sampling trace function. (See Section 5.9)

In the example above, 32 devices/registers data will be transferred to the latch data storage area when R0100 is changed from OFF to ON.

The latched data display screen is shown below.



This function is useful for program debugging.

5.11 Debug support function

The following functions are supported by T3 for effective program debugging.
(Refer to separate manuals for programmers for operation of these.)

5.11.1 Force function

There are two functions in the force function, input force and coil force. Batch input data is not updated in the input force specified register/device. The registers/devices which can be specified for forced input are the input register/device (XW/X), link register/relay (W/Z) in the receiver area and link register/relay (LW/L) in the receiver area. On the other hand, coil force specified coil instruction can not be processed when the program is running, so despite the state of the program, the coil device maintains its previous state.
Simulated input and simulated output are made possible by the combined use of the force function and the data setting function.

5.11.2 Online program changing function

This function enables to change the user program online (during RUN). The changes are made after completion of one scan, so it extends the inter-scan cycle.
Online program change is subject to the following conditions.

- You cannot make changes to the number or order of execution control instructions (below).
  END, MCS, MCR, JCS, JCR, JUMP, LBL, FOR, NEXT, CALL, SUBR, RET, IRET
- You cannot change the SFC structure in the SFC program section, but you can change the detail parts (ladder diagram) which relate to steps and transitions.

Also, there is the constant operand changing function.
This function enables to change the constant operand, such as timer/counter preset value and constant data used in function instructions, online (during RUN). For the timer/counter presets, changing is possible even in the memory protect state (P-RUN).

NOTE

When using the online program changing function, pay attention for safety.
If changed rung contains a transition-sensing type instruction (below), the instruction will be executed at the online changing if the input condition is ON, because the input condition of last scan is initialized. Pay attention for this point.
⊣↑⊢, ⊣P⊢, ⊣(P)⊢, Edged function instructions.

5.11.3 DEBUG mode functions

The T3 has a special mode for supporting the program debugging. It is the DEBUG mode. In the DEBUG mode, the following functions become available.

- Breakpoint setting function
  Starts and stops at the instruction which is set as the breakpoint.
- Single step execution function
  Starts and stops in unit of one instruction.
- Single rung execution function
  Starts and stops in units of one rung.
- N scans execution function
  Executes specified times of scans and stops.
- Stop condition setting function
  Executes until the specified stop condition is fulfilled.

DEBUG mode

The T3 can enter into the DEBUG mode only from the HALT mode. There are three sub-modes in the DEBUG mode, D-HALT, D-RUN and D-STOP.

D-HALT: When mode is changed from HALT to DEBUG, T3 enters this mode. The execution condition setting of the DEBUG mode function is possible in this mode. (All outputs OFF)

D-RUN: Program execution mode. When the stop condition is fulfilled in each DEBUG mode function, the mode moves into D-STOP.

D-STOP: Temporary stop mode. The mode transition factor of D-RUN to D-STOP can be displayed on the programmer. (Output state remains)

I/O disable

In the DEBUG mode, I/O module accessing can be disabled by the execution condition setting. When I/O disable is selected, external input status is not read into the input devices/registers (X/XW) and the status of the output devices/registers (Y/YW) is not sent the output modules.
In this case, operation modes displayed on the programmer are changed from D-HALT to S-HALT, D-RUN to S-RUN and D-STOP to S-STOP respectively.

Trace back function

In the program execution of the DEBUG mode functions, the online trace information of latest 10 scans is maintained. This information can be monitored after the execution is stopped (D-STOP mode).

*1) This function is not available for the single step execution and the single rung execution.
*2) This function is available only for the program range currently monitored.

Function details

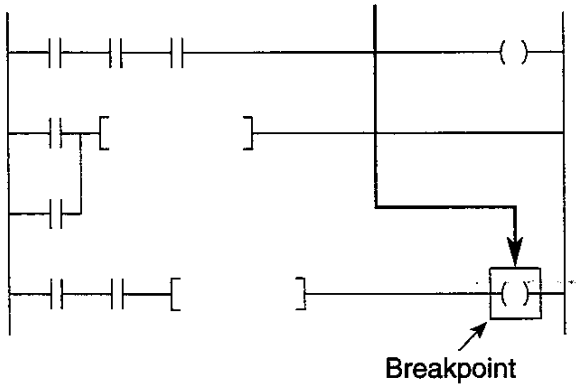
(1) Breakpoint setting function
Program execution is stopped when the instruction which is set as the breakpoint is fetched. The breakpoint can be set on one location only. This function becomes available when any number except 0 is set in the Breakpoint counts in the execution condition setting. When the breakpoint is fetched specified times, the program execution is stopped.

The start of execution can be selected from the initial start and the continue start.

- Initial start .....User data initialization is performed then program execution is started from the top.
- Continue start .....Program execution is started from the point where the execution was stopped last time.

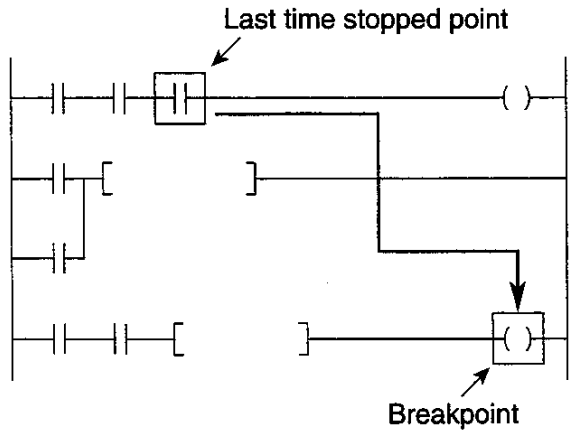
When execution is started from the D-HALT mode, the initial start is selected automatically.

Execution example 1 (Initial start)



User data initialization is performed. Then program execution is started from the top and stopped at the breakpoint. (The breakpoint instruction is not executed)

Execution example 2 (Continue start)



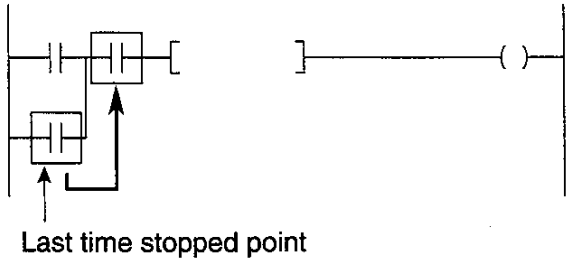
Execution is started from the point of last time stopped and stopped at the breakpoint.

(2) Single step execution function

The execution is started and stopped in units of one instruction. When this function is activated from the D-HALT mode, the user data initialization is performed and the program execution is stopped at the top instruction. (D-RUN → D-STOP)

When this function is activated from the D-STOP mode, T3 executes the last time stopped instruction and stops at the next instruction.

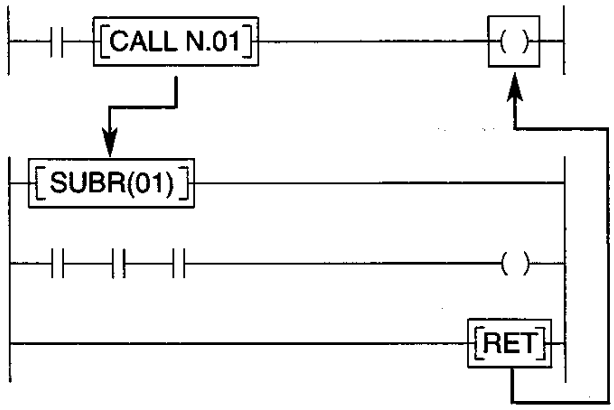
Execution example 1



Executes the last time stopped instruction and stops at the next instruction.

If execution is stopped at the sub-routine call instruction (CALL) and if the sub-routine call condition is satisfied, the next stop point is the corresponding sub-routine entry (SUBR).

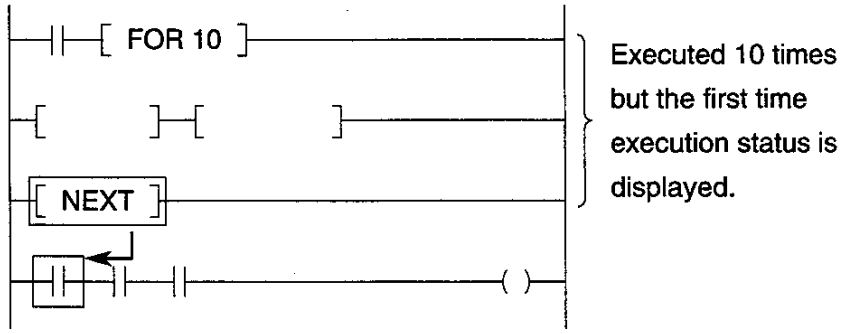
Execution example 2 (CALL/RET)



As same as above, if execution is stopped at the jump instruction (JUMP) and if the jump condition is satisfied, the next stop point is the corresponding label instruction (LBL).

In case of the FOR-NEXT loop, the instructions inside the loop are executed specified times, but the execution trace is not possible. The first time execution status is displayed and the execution is stopped at the next instruction to the loop.

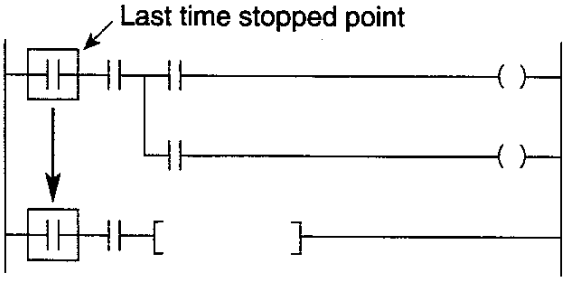
Execution example 3 (FOR-NEXT)



The interrupt program is executed during the single step execution, but it is not traced.

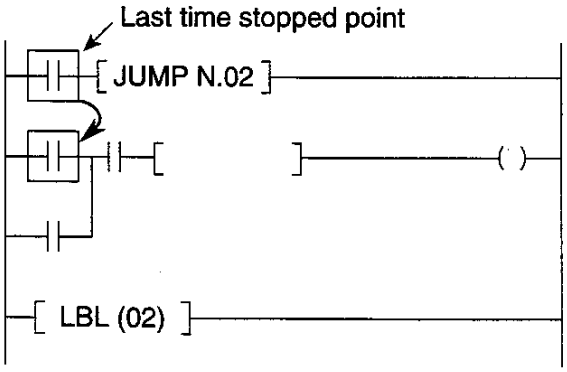
(3) Single rung execution function
The execution is started and stopped in units of one rung. When this function is activated from the D-HALT mode, the T3 performs the user data initialization and stops at the top instruction.
(D-RUN → D-STOP)
When this function is activated from the D-STOP mode, the T3 executes the last time stopped rung and stops at the first instruction of the next rung.

Execution example 1



Even if the rung contains the sub-routine call (CALL) or the jump (JUMP) instructions, the next stopping point is the next rung despite of calling or jumping.

Execution example 2 (JUMP)



If jump condition is not satisfied, the execution is stopped at the next rung.

If jump condition is satisfied, the execution is moved to the LBL instruction. (not stopped)

In case of the FOR-NEXT loop, the instructions inside the loop are executed specified times, but only the first time execution can be traced as same as the single step execution.

Also, the same precautions as the single step execution are applied to the interrupt program.

(4) N scans execution function
The T3 executes the specified times of scans and stops at the end of the scan.
The scan counts is set in the execution condition setting. The setting range is 0 to 65535. If 0 is set, this function is disabled.

The start of execution can be selected from the initial start and the continue start, as same as the breakpoint setting function.

(5) Stop condition setting function
The T3 executes the program until the stop condition is fulfilled.
The checkpoint of the condition can be selected either at the end of scan or at the breakpoint.

The stop condition can be set as either AND or OR conditions of up to four registers/devices data.

The start of execution can be selected from the initial start and the continue start, as same as the breakpoint setting function.

Notes (1) The DEBUG mode functions can also be used in combinations as follows.

(Breakpoint setting / or / Single step execution / or / Single rung execution) and/or (N scans execution / and/or / Stop condition setting)

(2) The initial load is not performed at the mode changing from D-HALT (S-HALT) to D-RUN (S-RUN).

(3) The timers used in the program are updated as normal in free scan, and updated as 100 ms/scan in the single step/rung execution.

(4) The sub-program execution is not interrupted in the single step/rung execution. In free scan, it is interrupted as normal.

(5) The actions of the interrupt program are as follows.

At D-HALT (S-HALT).....inhibited
At D-STOP (S-STOP) ...holded (executed when changed to enable)
At D-RUN (S-RUN) .....enabled

Restrictions (1) The DEBUG mode function is not available for the SFC program block.

(2) The DEBUG mode function is available only when the programmer is connected directly to the T3's programmer port.

(3) Program modification should not be made in the DEBUG mode. Otherwise, the DEBUG mode functions may not work correctly.

NOTE

In the D-STOP and D-RUN modes, FAULT LED blinks. And in the S-STOP and S-RUN modes, FAULT and I/O LEDs blink. Both of above are not error.

5.12 System diagnostics

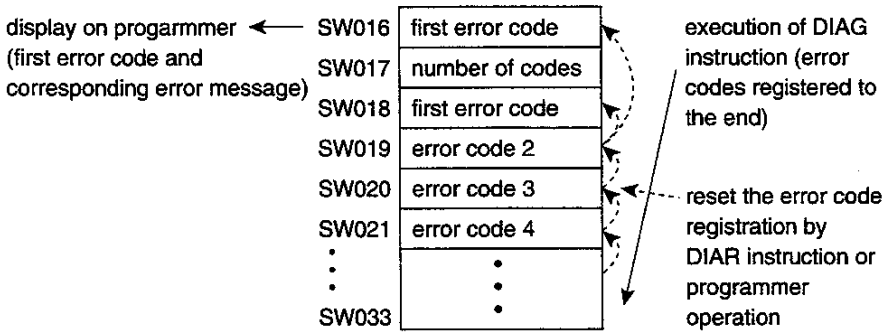
The following functions are provided for diagnosis of controlled system operation. The system can be monitored easily using of these functions.

(1) Diagnostics display function

By using the diagnostics display instruction (DIAG) in the user program, the relevant error code (1-64) and error message (maximum 12 characters per message) can be displayed on the programmer screen. Also, the error code generated is stored in the special registers (SW016-SW033) in order of generation up to a maximum of 16 codes and the annunciator relay (S0340-S037F) corresponding to the error code goes ON. It is possible to use the special register/relay to display the error code on an external display monitor.

The error codes registered can be reset one by one (shift up after erased) using the programmer or by the diagnostics display reset instruction (DIAR).

This function may also be used effectively in conjunction with the bit pattern check and the sequence time over detection mentioned below. (Refer to details of diagnosis display instructions in other manual for instruction set)



When error codes are registered, for example 3, 10, 29, 58, each corresponding annunciator relay, S0342, S0349, S035C, S0379 comes ON.

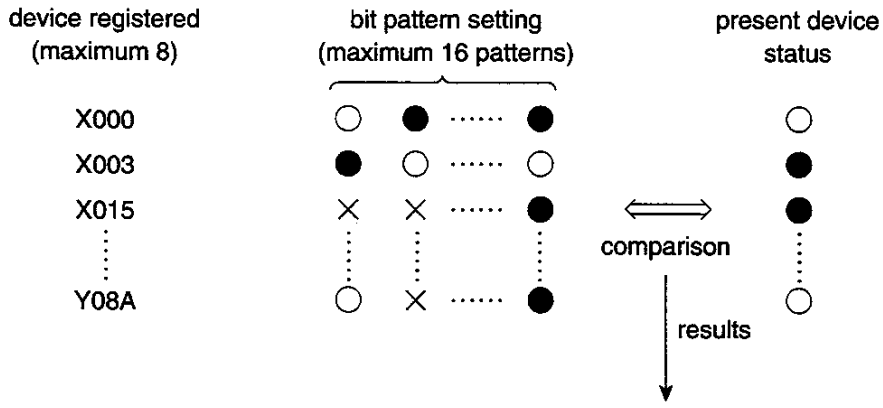
(Annunciator relay)

	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
SW034	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
SW035	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
SW036	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33
SW037	64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49

(2) Bit pattern check function

This function checks that the device ON/OFF status for a number of devices are in the normal combinations (pattern). For example, checks that not more than 2 from device 1, 2 and 3 are ON simultaneously. Up to 8 devices can be registered, and up to 16 patterns can be set. The checkpoint can be selected either before program execution or end of scan. The results are reflected in the special relay S0142.

This function is enabled when the special relay S0140 is set to ON.



agrees with the setting pattern .....S0142 =0
dose not agree with the setting pattern ...S0142 =1

In the pattern setting, OFF is shown as ○, ON is shown as ● and do not care is shown as ×.

The device and bit pattern registration takes place in programmer system diagnosis menu.

*) The checkpoint of this function can be selected by the special relay S015F as below.

S015F = OFF .....before user program execution
(after I/O processing)

S015F = ON .....after user program execution

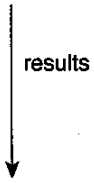
(3) Register value validity check function

This function checks that the register value is within the specified numerical value range. Up to 4 registers can be registered with the maximum and the minimum data. Also, it is possible to select the register value to be taken as an integer (signed) or as a positive integer (unsigned).

The checkpoint can be selected either before program execution or end of scan. The results are stored in the special relay S0143-S0146 (within the range: 0, outside the range: 1).

This function is enabled when the special relay S0140 is set to ON.

registered register (maximum 4)	type	minimum value	maximum value		present register value
XW034	unsigned	0	400		200
XW035	signed	-1500	1500	⟺	2000
D0011	unsigned	H0200	H9000	comparison	H1234
W0100	signed	-300	600		-1000



register 1 (XW034) ... S0143 = 0
register 2 (XW035) ... S0144 = 1
register 3 (DO011) ... S0145 = 0
register 4 (WO100) ... S0146 = 1

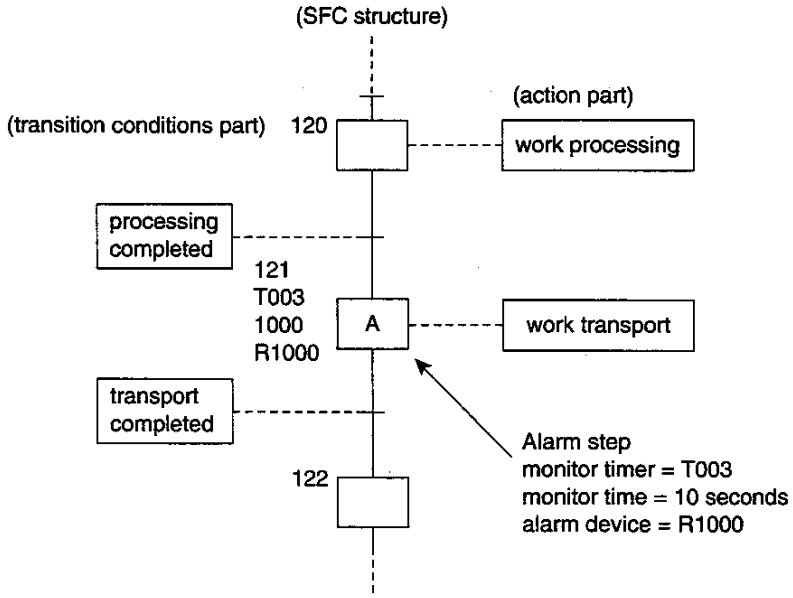
The register and the numerical value range are registered in programmer system diagnosis menu.

*) The checkpoint of this function can be selected by the special relay S015F as below.

S015F = OFF .....before user program execution
(after I/O processing)

S015F = ON .....after user program execution

(4) Sequence time over detection function
The alarm step is provided for one of SFC (sequential function chart) instructions. This Alarm step turns ON the specified device when the following transition is not come true within the preset time. This function allows easy detection of operation hold ups in sequential control process



With the above example, if the transport has not been completed (work arrived signal ON etc) within 10 seconds from when the work transport started, the specified alarm device (R1000) comes ON. By this means a malfunction of the work drive or the sensor can be detected.

Refer to Part 3 of this manual and the other instruction set manual for explanation with respect to SFC.

5.13 Password function

For the system security, the password function is provided.
There are three levels of protection as shown below. Accordingly, three levels of passwords can be set.

Level 1 possible functions

- Clear memory
- Writing/down-loading program
- Operation mode control
- Setting/changing password

Level 2 possible functions

- Reading/up-loading program
- Program write to IC memory card (EEPROM)

Level 3 possible functions

- Writing data
- Writing system information
- I/O allacation
- Sampling trace, status latch

Always possible functions

- Reading system information
- Reading I/O allocation information
- Reading event history
- Reading data

For example, if level 1 and level 2 passwords have been set, only level 3 and always possible functions are enabled. In this state, if the level 2 password is entered, the level 2 possible functions are also enabled.

NOTE

(1) Do not forget your level 1 password. Otherwise, you cannot release the password protection.
(2) Protection level for each programmer command is explained in the programmer operation manual.

PART 3
PROGRAMMING INFORMATION

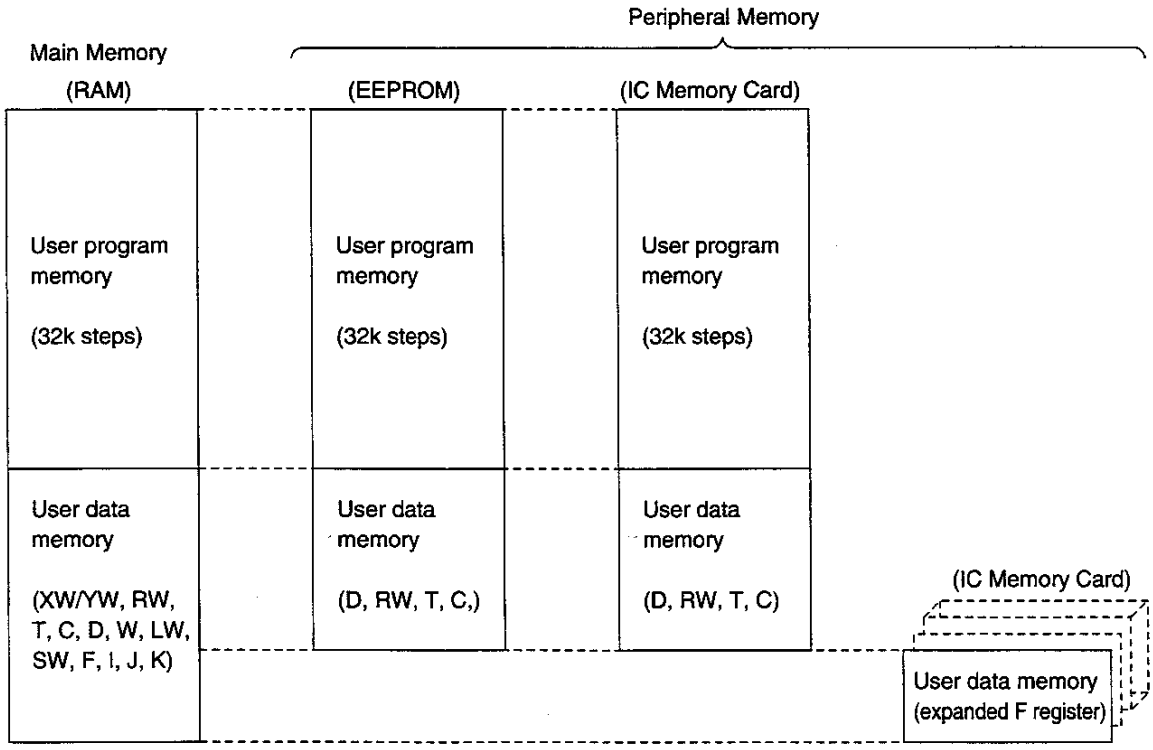
1.1 Aims of Part 3

The main functions of the T3 are to store the user program, to execute the stored user program and to control and monitor the operation/state of machines/processes which are the result of such execution.
The user program is a series of instructions for achieving the request control function, operation conditions, data processing and the interface with the operator. It is stored in the user program memory. The execution of the user program is the sequential performance of the processes of reading user data in which external input/output data and control parameters are stored, processing the respective instructions and storing the results of this in the user data memory.

Part 2 described the types of processing which are executed by the T3 internally, functions for executing the user program efficiently and the RAS functions. Part 3 describes the necessary information for creating user programs, that is to say detailed user data, detail of the input/output allocation and the programming languages. Also, the user program configuration is described to use the T3's multi-tasking function.

1.2 User memory configuration

The following diagram shows the user memory configuration of the T3.



The memory which can be used by user is called user memory. The user memory can be divided by configuration into main memory and peripheral memory. And the user memory can be devided by function into user program memory and user data memory.

The main memory is a built-in RAM memory with battery backed up. On the other hand, the peripheral memory is an optional memory configured by IC memory card or EEPROM. The peripheral memory can be used as back up for main memory (user program and register data) or expansion memory (expanded file register, IC memory card only).

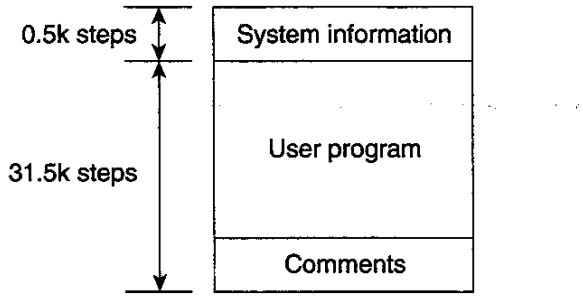
The user program memory has a capacity of 32k steps (step is a unit for instruction storage), and stores a series of instructions created by ladder diagram or SFC.

The user data memory stores variable data for user program execution. It is separated by function into input/output registers, data registers, etc.

2.1 Overview

The user program memory can be divided into the system information storage area, the user program storage area and comments storage area as shown below.

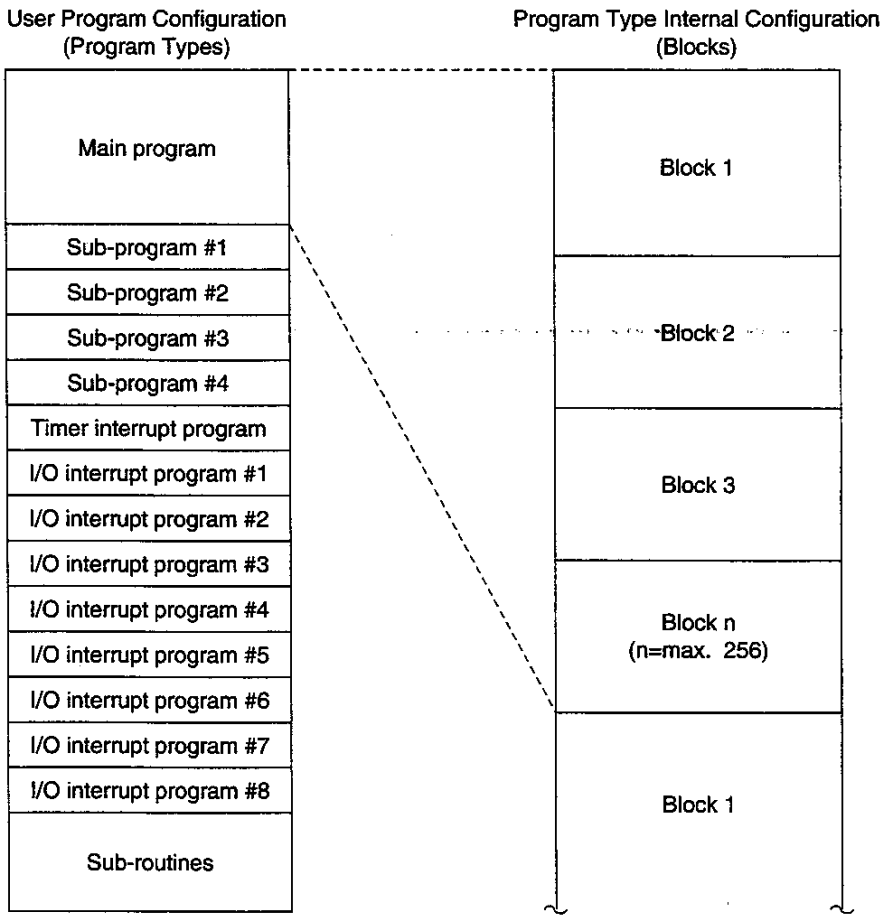
User Program Memory Configuration



System information is the area which stores execution control parameters for the user program and user program management information, and it always occupies 0.5k steps.

Comments are added and stored for easy maintenance of the user program. The comments storage area is not fixed. (user setting)

The user programs is divided into the program types of main program, sub-programs, interrupt programs and sub-routines, depending on the function.
Of these, the main program is the core of the user program.
On the other hand, when it is difficult to achieve the requested control functions by the main program alone, sub-programs and interrupt programs are used as required, but need not be provided.
Also, sub-routines are used when repetition of the same process in a program is required, or in order to see the program more easily by making one function into a block, but may not be provided if not required.



Also, in each program type, the user program is arranged by units called 'blocks'.

Internally, a block definition label is present at the head of each block. The program type, block number and programming language information are in the block definition label (there is no need for the user to be concerned with the block definition label).

Although the 2 programming languages of ladder diagram and SFC can be used in combination in the T3, only 1 language can be used in any 1 block.

NOTE

(1) In each program type and block, there is no limit to the program capacity (number of steps). The only limit is the total capacity (31.5k steps).
(2) The block number need not be consecutive. In other words, there may be vacant blocks in the sequence.

2.2 System information

System information is the area which stores execution control parameters and user program management information when executing a user program, and occupies 0.5k steps of the user program memory. The following details are included in system information.

(1) Program ID
This is the user program identification. A setting of up to 10 alphanumeric characters can be set. The program ID can be registered/monitored on the system information screen of the programmer.

(2) System Comments
These are comments attached to the user program. A setting of up to 30 alphanumeric characters can be set. The system comments can be registered/monitored on the system information screen of the programmer.

(3) Memory Capacity
This stores the memory type (user program capacity/data register capacity). The memory capacity can be monitored on the system information screen of the programmer. (monitor only)

(4) Steps Used
This stores the number of steps used in the user program. The number of steps used can be monitored on the system information screen of the programmer. (monitor only)

(5) PLC Type
This stores the model type. The PLC type can be monitored on the system information screen of the programmer. (monitor only)

(6) Program Size Setting
This is the capacity assigned to the user program. The rest of this setting out of total 32k steps is assigned to the comments. The program size setting can be registered/monitored on the system information screen of the programmer.

(7) Sampling Buffer Setting
This performs the setting and registration of the storage capacity of the sampling buffer for the sampling trace function. The maximum setting is 8k words. The sampling buffer setting can be registered/monitored on the system information screen of the programmer.

(8) Retentive Memory Area Designation
This sets and registers the address ranges for the auxiliary register (RW), timer register (T), counter register (C) and data register (D) which retain pre-power cut data out of the user data. The ranges registered here are outside the subjects of the user data initialization process. For each of these registers, the ranges from the leading address (0) to the designated address are the retentive memory areas. The retentive memory area designations can be registered/monitored on the system information screen of the programmer.

(9) Scan Time Setting
This sets and registers the scan mode (floating/constant). When no scan time is registered (blank), the mode becomes the floating scan mode. When a numerical value is set for the scan time, the mode becomes a constant scan mode which takes that time as the scan cycle. The setting for the scan cycle is 10-200 ms (in 10 ms units).
The scan time setting can be registered/monitored on the system information screen of the programmer.

(10) Sub-Program Execution Time
Time limit factor assigned for sub-programs in the floating scan. The setting range is 1-100 ms (in 1 ms units). The sub-program execution time can be registered/monitored on the system information screen of the programmer.

(11) Timer Interrupt Interval
This sets and registers the interrupt cycle of the timer interrupt program. The setting range is 2-1000 ms (in 1 ms units). The timer interrupt interval can be registered/monitored on the system information screen of the programmer.

(12) Computer Link Parameters
This sets and registers the parameters for the computer link. The computer link parameters can be registered/monitored on the system information screen of the programmer.

The parameter items and their setting ranges are as follows:
* Station No. .....1-32 (initial value=1)
* Baud rate .....300, 600, 1200, 2400, 4800, 9600, 19200(initial value 9600)
* Parity.....None, odd, even (initial value=odd)
* Data length (bits).....7, 8 (initial value=8)
* Stop bit.....1, 2 (initial value=1)

(13) I/O Allocation Information
This stores I/O allocation information and unit base address designation information. This information is created either by executing the automatic I/O allocation command or by setting and registering an I/O module type for each slot (manual I/O allocation) on the I/O allocation information screen of the programmer.

(14) Interrupt Assignment Information
This stores the information of correspondence between the I/O interrupt program and I/O modules with interrupt functions. In the initial state (without setting this information), the lower number of I/O interrupt programs are assigned in sequence from the interrupt module closest to the CPU.
This information can be registered/monitored on the interrupt assignment screen of the programmer.

(15) Network Assignment Information
Information on the link register areas allocated to the data transmission modules (TOSLINE-S20, TOSLINE-F10) is stored here. This information can be registered/monitored on the network assignment information screen of the programmer.

2.3 User program

The user program is composed of each of the program types of main program, sub-programs (#1 - #4), interrupt programs (Timer, I/O#1 - I/O #8) and sub-routines. Of these program types, a main program must always be present. However, the other program types may not be present at all if they are not used. Therefore, needless to say, a user program can be configured with a main program only.

Also in the program types, the program can be divided into units called 'blocks' (block division is not necessary unless required). Block division is required in the following cases.

* When using languages other than ladder diagram (1 language/ block)
* When creating multiple SFC programs (1 SFC/block, see Section 5.3)
* When block division by control function units makes the program easier to see.

There are no restrictions on program capacities (number of steps) by program types and blocks. (Except in the case of SFC)

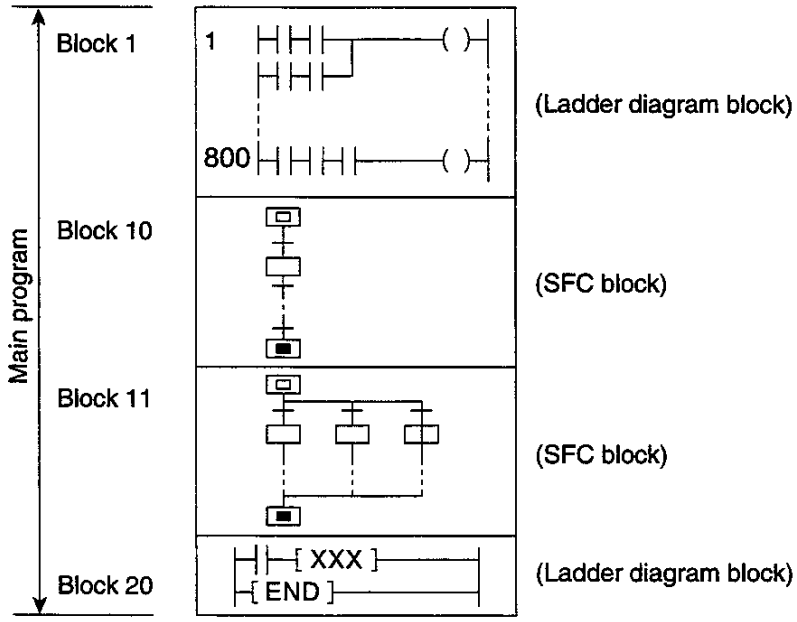
As block numbers, 1 to 256 are available. However, the block numbers need not be consecutive. When executing the program, the program is executed in sequence from the block with the lowest number.

2.3.1 Main program

The main program is the portion which is the core of the user program and is always executed every scan. The main program must be finished by the END instruction.

Although instructions may be present after the END instruction, these portions will not be executed. (However, they count in the number of steps used)

(Example of Main Program Configuration)



2.3.2 Sub-program

The sub-program is a program type to achieve the multi-tasking function. 4 sub-programs (Sub #1 - Sub #4) are provided.

Sub #1 is executed once in the first scan before the main program execution. Therefore, the Sub #1 can be used for the initial setting program.

Sub #2 can be selected from the two functions, the initial setting program in the case of power interruption and the normal sub-program function which can be controlled by other program types.

Sub #3 and Sub #4 are fixed as the normal sub-program function.

In the normal sub-program function of Sub #2, Sub #3 and Sub #4, the execution mode can be selected either the one time mode or the cyclic mode.

NOTE

For the details of the sub-program execution, see Part 2 Section 3.2. Also, for Sub #2, see Part 2 Section 5.5.2.

Each sub-program must be finished by the END instruction.
Although instructions may be present after the END instruction, these instructions will not be executed. (However, they count in the number of steps used)

Sub-programs execution conditions are summarized in the table below.

Sub No.	Execution condition
Sub #1	Executed once in the first scan before the main program execution, except when T3 is in the hot restart mode (S0400=1 and power recovery within 2s).
Sub #2	[Special mode] S0403=1 Executed once in the first scan before the main program execution when T3 is in the hot restart mode (S0400=1 and power recovery within 2s).
	[One time mode] S0403= 0 and S0405=0 Executed once when S0409 is changed from 0 to 1. (S0409 is reset to 0 automatically)
	[Cyclic mode] S0403=0 and S0405=1 Executed once per every specified number of scans which is specified by SW042, during S0409=1.
Sub #3	[One time mode] S0406=0 Executed once when S040A is changed from 0 to 1. (S040A is reset to 0 automatically)
	[Cyclic mode] S0406=1 Executed once per every specified number of scans which is specified by SW043, during S040A=1.
Sub #4	[One time mode] S0407=0 Executed once when S040B is changed from 0 to 1. (S040B is reset to 0 automatically)
	[Cyclic mode] S0407=1 Executed once per every specified number of scans which is specified by SW044, during S040B=1.

NOTE

The sub-program execution may be time-sliced by scan. Therefore, to prevent the unexpected status changes of I/O registers (XW/YW) used in the sub-program, it is recommended to use the batch I/O inhibition (with i allocation) and the direct I/O instruction (I/O).

2.3.3 Interrupt program

There are a total of 9 types of interrupt program. These are 1 timer interrupt program which is executed cyclically with a cycle which is set in system information, and 8 I/O interrupt programs (#1 - #8) which are started by interrupt signals from I/O modules with interrupt function.

- Timer interrupt program
  This is executed cyclically with a cycle of 2-1000 ms which is registered in system information. When no cycle is registered (blank), it is not executed.
  Set the interval setting of the timer interrupt with 1 ms units in item 16 of the T-PDS system information screen.
  For details, see T-PDS operation manuals.

- I/O interrupt programs (#1 - #8)
  These are started by interrupt signals generated by I/O modules with the interrupt function. The coordination between the interrupt program numbers and the I/O modules with interrupt function can be changed by the interrupt assignment function.

Each interrupt program must be finished by the IRET instruction.

NOTE

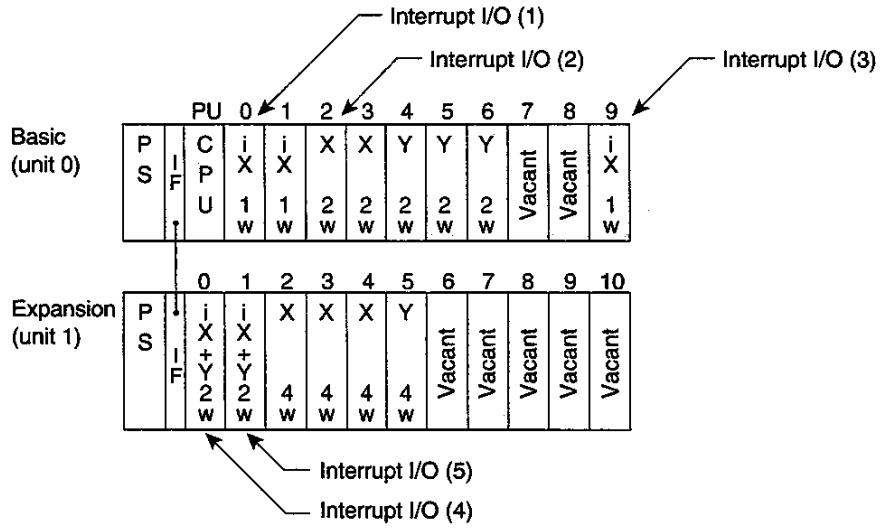
(1) For details of interrupt program operation, see Part 2 Section 3.3.
(2) SFC cannot be used in the interrupt program.

The following modules are available as the I/O module with the interrupt function (interrupt I/O).

- Change detect 8 pts DC input
  (Part No.: CD 332, allocation type: iX1W)
- 2 channels pulse input
  (Part No.: PI312, allocation type: iX+Y2W)

When automatic I/O allocation is carried out in the state with interrupt I/O mounted, for coordination between the interrupt program number and the interrupt I/O, the lower number I/O interrupt programs are allocated in sequence from the interrupt I/O closest to the CPU. (See the example on the following page)

Example)
(1) Module mounting status



(2) Register allocation

Unit 0

Slot	Module Type	Register
PU	-	-
0	iX 1W	XW000
1	iX 1W	XW001
2	X 2W	XW002, XW003
3	X 2W	XW004, XW005
4	Y 2W	YW006, YW007
5	Y 2W	YW008, YW009
6	Y 2W	YW010, YW011
7	Vacant	-
8	Vacant	-
9	iX 1W	XW012

Unit 1

Slot	Module Type	Register
0	iX+Y 2W	XW013, YW014
1	iX+Y 2W	XW015, YW016
2	X 4W	XW017~XW020
3	X 4W	XW021~XW024
4	Y 4W	YW025~YW028
5	Y 4W	YW029~YW032
6	Vacant	-
7	Vacant	-
8	Vacant	-
9	Vacant	-
10	Vacant	-

(3) Interrupt program assignment

Program type	Corresponding input register	Correponding interrupt I/O	Remarks
I/O interrupt program #1	XW000	Unit 0-Slot 0	Interrupt I/O (1)
I/O interrupt program #2	XW001	Unit 0-Slot 1	Interrupt I/O (2)
I/O interrupt program #3	XW012	Unit 0-Slot 9	Interrupt I/O (3)
I/O interrupt program #4	XW013	Unit 1-Slot 0	Interrupt I/O (4)
I/O interrupt program #5	XW015	Unit 1-Slot 1	Interrupt I/O (5)

The interrupt program assignment determined as the page before can be changed as follows.

Example)

Interrupt assignment information (before changing)

Interrupt level	Interrupt program No.	Input register No.
0	[1]	XW000
1	[2]	XW001
2	[3]	XW012
3	[4]	XW013
4	[5]	XW015

Change to

Interrupt assignment information (after changing)

Interrupt level	Interrupt program No.	Input register No.
0	[1]	XW000
1	[2]	XW001
2	[3]	XW012
3	[5]	XW013
4	[4]	XW015

In this example, interrupt programs for XW013 and XW015 are exchanged.

NOTE

By using the interrupt assignment function, the correspondence between the interrupt I/O and the interrupt program No. can be changed. However, the interrupt level (priority) is fixed as the hardware. The interrupt I/O mounted closer to the CPU has higher interrupt priority. The interrupt priority cannot be changed.

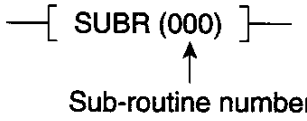
2.3.4 Sub-routines

When it is necessary to execute repetitions of the same process in a program, this process can be registered as a sub-routine. This sub-routine can be executed by calling it at the required location. By this means, the number of program steps can be reduced and, at the same time, the program becomes easier to see since the functions have been put in order.

Sub-routines can be called from other program types (main program, sub-programs, interrupt programs) and from other sub-routines (they can also be called from the action part of SFC).

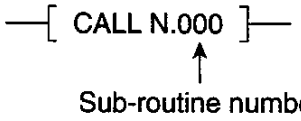
The sub-routine should be located in the program type "Sub-routine", and started by SUBR instruction and finished by RET instruction. Up to 256 sub-routines can be programmed.

It is necessary to assign a sub-routine number to the SUBR instruction (sub-routine entry instruction). The effective numbers are from 0 to 255.

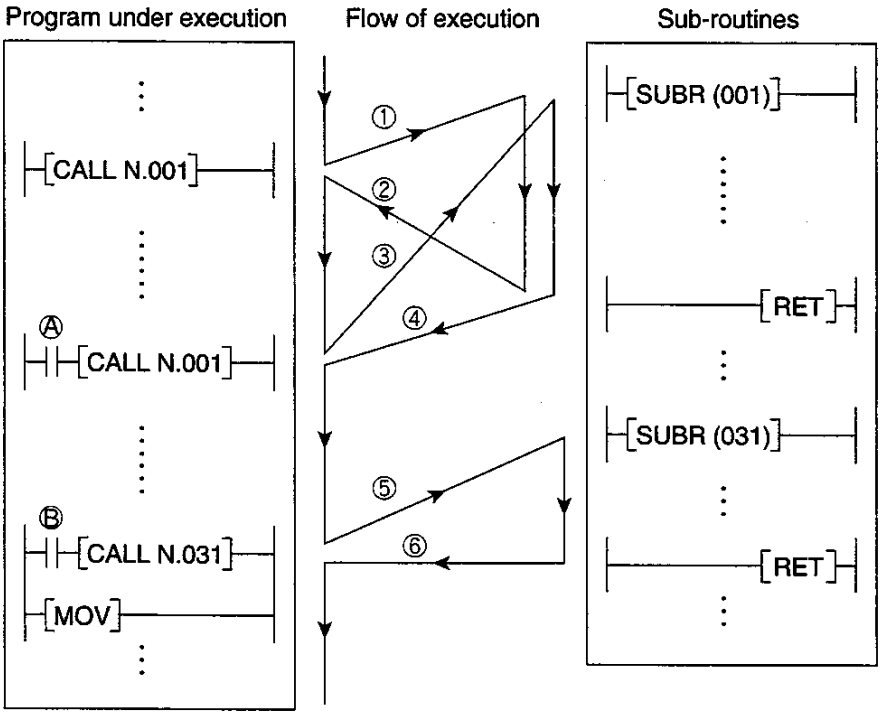


The RET instruction (sub-routine return instruction) has no sub-routine number.

The instruction which calls a registered sub-routine is the CALL instruction (sub-routine call instruction) of ladder diagram. The CALL instruction requires the number of the sub-routine it calls.



The following is an execution sequence when sub-routines are included.



① By the sub-routine 001 CALL instruction execution, the execution shifts to sub-routine 001
② When it has proceeded to the RET instruction, the execution returns to the instruction following the CALL instruction in ①
③ When device Ⓐ is ON, the CALL instruction is executed, and the execution shifts to sub-routine 001
④ When it has proceeded to the RET instruction, the execution returns to the instruction following the CALL instruction in ③
⑤ When device Ⓑ is ON, the CALL instruction is executed, and the execution shifts to sub-routine 031
⑥ When it has proceeded to the RET instruction, the execution returns to the instruction following the CALL instruction in ⑤ (the MOV instruction in this example)

NOTE

(1) Multiple sub-routines can be programmed in a block. However for execution monitor by programmer, 1 sub-routine on 1 block is recommended.
(2) SFC cannot be used in a sub-routine.
(3) Other sub-routines can be called from a sub-routine (nesting), up to 6 layers.
(4) Since the operation will become abnormal in cases such as calling the same sub-routine during the execution of a sub-routine, take care that the cases do not occur.

2.4 Comments

Comments can be added and stored in the T3's user program memory. By this means, the user program becomes easier to understand.

The types of comments which can be stored in the T3 are tags/comments for registers, devices and SFC steps.

Tag.....up to 5 characters
Comment.....up to 20 characters

The comments storage capacity is the rest of the program size setting out of total 32k steps.
The maximum storage number of comments (tag and comment paired) is calculated as follows.

$$(1024 \times (32 - N) - 38) / 10$$

N: Program size setting (assigned to the user program)

NOTE

Here, the comments which can be stored in the T3 are explained. Comments can also be saved in a disk file. For the disk file usage, see separate manual for the programmer (T-PDS).

3.1 Overview

The area which stores the external input/output data, current values of timers and counters and the values of the variables for data processing is called the 'user data'.

For user data, the storage location of the data is expressed by a combination of 'function type' and a sequence of numbers which starts from 0 (this is called the 'address')

Example) XW 005

- 005 — Address 005 (in this case it is the register address)
- XW — Function type XW=Input register

To say that the content of XW005 is 100 is to say that the numerical value 100 is stored in a location in the user data memory indicated by XW005.

Also, user data is divided into registers and devices according to the type of data to be stored. (Although the expression 'relay' is also used, a relay should be regarded as one type of device)

A 'register' is an area which stores 16 bits of data and it is expressed as a combination of a function type and a register address.
(the register address is a decimal number)

Example) D 1024

- 1024 — Register address (decimal number)
- D — Function type D=Data register

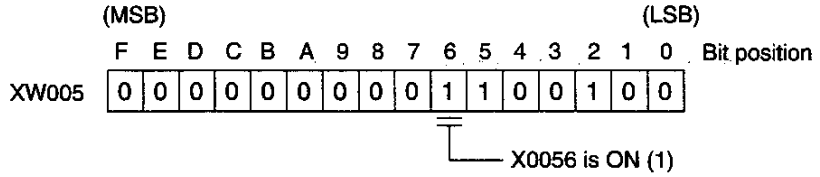
On the other hand a 'device' is an area which stores 1 bit of data (it expresses 1 or 0, in other words ON or OFF), and it is expressed as a combination of a function type and a device address. However, a device does not use an independent memory area. It is allocated as 1 bit in the 16 bits of the corresponding register. Therefore, the device address is expressed in the form of the corresponding register address+bit position.

Example) X 005 6

- 6 — Bit position (there are 16 positions 0-F)
- 005 — Register address (decimal number)
- X — Function type X=Input device (corresponds to input register XW)

The correspondence between register data and device data should be considered as follows.

Example) When it is said that the content of XW005 is 100, since the decimal number 100 is expressed as 1100100 in binary notation, this indicates that each of the bits of XW005 will be as follows.



At this time, the data of device X0056 corresponding to bit position "6" of XW005 is 1, that is to say X0056 is ON.

The correspondence of registers and devices is shown by function types.

- Input device (X) .....corresponds to 1 bit of input register (XW)
- Output device (Y).....corresponds to 1 bit of output register (YW)
- Auxiliary device (R).....corresponds to 1 bit of auxiliary register (RW)
- Special device (S).....corresponds to 1 bit of special register (SW)
- Link device (Z).....corresponds to 1 bit of link register (W) (but only in the leading 512 words)
- Link relay (L).....corresponds to 1 bit of link register (LW)

The treatment of the other devices, I, O, T. and C., is slightly different. It is described in detail in Section 3.2.

The following Table shows the types of registers and devices and their address ranges. Their functions and methods of use are described in Section 3.2.

Function Type	Type Code	Address Range	Quantity	Expression Example
Input register	XW	000-255	Total 256 words	XW001
Output register	YW			YW034
Direct input register	IW			IW001
Direct output register	OW			OW034
Input device	X	0000-255F	Total 4096 points	X001A
Output device	Y			Y0348
Direct input device	I			I0012
Direct output device	O			O0340
Auxiliary register	RW	000-511	512 words	RW100
Auxiliary device	R	0000-511F	8192 points	R1001
Special register	SW	000-255	256 words	SW014
Special device	S	0000-255F	4096 points	S0140
Timer register	T	000-511	512 words	T030
Timer device	T.	000-511	512 points	T.030
Counter register	C	000-511	512 words	C199
Counter device	C.	000-511	512 points	C.199
Data register	D	0000-8191	8192 words	D4055
Link register	W	0000-1023	1024 words	W0200
Link device	Z	0000-511F	8192 points	Z2001
Link relay register	LW	0000-255	256 words	LW123
Link relay	L	0000-255F	4096 points	L123F
File register	F	0000-8191	8192 words	F0500
Index register	I	None	1 word	I
	J	None	1 word	J
	K	None	1 word	K

NOTE

In the T3, 1 word is treated as equal to 16 bits, and the number of registers is counted in word units.

3.2 Registers and devices

The following Tables describe the functions and address ranges for each function type of registers and devices

Input registers and Input devices

Codes	Input registers.....XW Input devices.....X
Addresses	Input registers.....000-255 (256 words) Input devices.....0000-255F (4096 points) } Common use as output registers/output devices
Functions	These are allocated in the input module as register units (word units) by perfoming input/output allocation. The signal state inputted to the input module is stored in the corresponding input register by batch input/output timing (except for modules which have the designation i attached when allocating). An input device expresses 1 bit of the corresponding input register. The data of input register/input devices basically do not change during 1 scan. However, when executing a direct I/O instruction (FUN235), data is read from the corresponding input module when the instruction is executed and is stored in an input register/input device (XW/X). Thus, the data changes during the scan.

Output registers and Output devices

Codes	Output registers.....YW Output devices.....Y
Addresses	Output registers.....000-255 (256 words) Output devices.....0000-255F (4096 points) } Common use as input registers/input devices
Functions	These are allocated in the output module as register units (word units) by performing input/output allocation. The data stored in the output register is written to the corresponding output module by batch input/output timing, and the state of the output signal of the output module is determined (except for modules which have the designation i attached when allocating). An output device expresses 1 bit of an output register.

Direct input registers and Direct input devices

Codes	Direct input registers.....IW Direct input devices.....I
Addresses	Direct input registers.....000-255 (correspond to input registers (XW)) Direct input devices.....0000-255F (correspond to input devices (X))
Functions	Direct input registers/direct input devices do not themselves indicate specific memories. When the instruction which uses these registers/devices is executed, they operate and read data directly from the input module corresponding to the address. These registers/devices are used when using the T3 as the direct input/output system (direct system) and not the batch input/output system (refresh system). Example) I0000 ⊣ ⊢ NO contact instruction of I0000 When executing the instruction, the bit data corresponding to X0000 is read from the input module and the instruction is executed by this data. (The X0000 data is not affected) -[IW005 MOV RW 100]- Transfer instruction from IW005 to RW100 When executing the instruction, the word data corresponding to XW005 is read from the input module and is transferred to RW100. (The XW005 data is not affected)

Direct output registers and Direct output devices

Codes	Direct output registers.....OW Direct output devices.....O
Addresses	Direct input registers.....000-255 (correspond to output registers (YW)) Direct input devices.....0000-255F (correspond to output devices (Y))
Functions	When instructions are executed using direct output registers/direct output devices, data is stored in the corresponding output registers/output devices (YW/Y). Then, this output register (YW) data is written directly to the corresponding output module. These registers/devices are used when using the T3 as the direct input/output system (direct system) and not the batch input/output system (refresh system). Example) O0020 -( )⊣ Coil O0020 When the instruction is executed, the data (ON/OFF data) corresponding to the left link state is stored in Y0020. Then the 16-bit data of YW002 is written to the corresponding output module.

Auxiliary registers and Auxiliary devices

Codes	Auxiliary registers.....RW Auxiliary devices.....R
Addresses	Output registers.....000-511 (512 words) Output devices.....0000-511F (corresponding to one bit in a register, 8192 points)
Functions	These are general purpose registers/devices which can be used for temporary storage of execution results in a program. An auxiliary register is used for storing 16-bit data. An auxiliary device indicates 1 bit in an auxiliary register. Auxiliary registers/devices can be designated as retentive memory areas.

Special registers and Special devices

Codes	Special registers.....SW Special devices.....S
Addresses	Special registers.....000-255 (256 words) Special devices.....0000-255F (corresponding to one bit in a register, 4096 points)
Functions	These are registers/devices which have special function such as fault flags (Error down/Warning) which are set when the CPU detects a malfuction; timing relays and clock calendar data (year, month, day, hour, minute, second, day of week) which are updated by the CPU; flags/data which the user sets for executing operational control of the sub-programs. For details, see the following table.

Timer registers and Timer devices

Codes	Timer registers.....T Timer devices.....T.
Addresses	Timer registers.....000-511 (512 words) Timer devices.....000-511 (512 points)
Functions	The timer registers are used together with timer instructions (TON, TOF, SS, TRG), and store elapsed time (increment system) when the timer is operating. Also, the timer devices are linked to the operation of the timer registers with the same address, and store the output results of timer instructions. T000 to T063 works as 0.01 sec timers and T064 to T511 works as 0.1 sec timers. The timer registers can be designated as retentive memory areas.

Counter registers and Counter devices

Codes	Counter registers.....C Counter devices.....C.
Addresses	Counter registers.....000-511 (512 words) Counter devices.....000-511 (512 points)
Functions	The counter registers are used together with counter instructions (CNT, U/D), and store the current count value when the counter is operating. Also, the counter devices are linked to the operation of the counter registers with the same address, and store the output results of counter instructions. The counter registers can be designated as retentive memory areas.

Data registers

Code	D
Addresses	0000-8191(8192 words)
Functions	General-purpose registers which can be used for such purposes as a temporary memory for arithmetic results and the storage of control parameters. Apart from the fact that bit designation is not possible, they can be used in the same way as auxiliary registers. Data registers can be designated as retentive memory areas. Also, when a peripheral memory is used, D0000-D4095 become subjects for the initial load. In the 'memory protect' state (P-RUN), data writing to D0000-D4095 is prohibited.

Link registers and Link device (TOSLINE-S20)

Codes	Link registers.....W Link devices.....Z
Addresses	Link registers.....0000-1023 (1024 words) Lind devices.....0000-511F (corresponding to the leading 512 words of the register, 8192 points)
Functions	Used for a data link by the TOSLINE-S20. For the leading 512 words (W0000-W0511) of the link registers, bit designation is possible as link devices (Z0000-Z511F). For areas not allocated to TOSLINE-S20, they can be used in the same way as auxiliary registers and data registers.

Link registers and Link relays (TOSLINE-F10)

Codes	Link registers.....LW Link relays.....L
Addresses	Link registers.....000-255 (256 words) Link relays.....0000-255F (4096 points)
Functions	Used as registers/relays for remote I/O by the TOSLINE-F10. When TOSLINE-F10 is not used, they can be used in the same way as auxiliary relays.

File registers

Code	F
Addresses	0000-8191 (8192 words)
Functions	Can be used in the same way as data registers for such as storing control parameters and storing field collection data. Bit designation is not possible. The whole file register area is retained for power off. The file registers can also be used for the sampling buffer.

Index registers

Codes	I, J, K (3 types, 3 words)
Addresses	None
Functions	When registers (apart from index registers) are used by instructions, apart from the normal address designation system (direct address designation, for instance D0100), indirect designation (indirect address designation, for instance D0100.I) is possible by using the index registers. (If, for instance the content of I is 5, D0100.I indicates D0105) For indirect address designation, see Section 3.4.

Tables of special registers/special relays are shown below.

Overall map

Register	Content
SW000	Operation mode, error flags, warning flags
SW001	CPU error-related flags
SW002	I/O error-related flags
SW003	Program error-related flags, IC memory card status
SW004	Timing relays
SW005	Carry flag, error flag
SW006	Flags related to error during program execution
SW007 ~ SW013	Clock-calendar data (Year, month, day, hour, minute, second, day of the week)
SW014	Flags related to bit pattern check/data validity check
SW015	Flags related to I/O error mapping, etc.
SW016 ~ SW033	Diagnosis display record (system diagnosis)
SW034 ~ SW037	Annunciator relay (system diagnosis)
SW038	Reserve (for future use)
SW039	Interrupt program execution status
SW040	Sub-program execution control
SW041	Sub-program execution status
SW042 ~ SW044	Sub-program execution intervals (for cyclic mode)
SW045	Power interruption continuous operation time
SW046 ~ SW049	I/O error map
SW050 ~ SW077	Reserve (for future use)

Overall map (continued)

Register	Content
SW078 ~ SW093	TOSLINE-F10 commands/status
SW094 ~ SW109	TOSLINE-F10 scan error map
SW110	TOSLINE-S20 CH1 station status
SW111	TOSLINE-S20 CH2 station status
SW112 ~ SW115	TOSLINE-S20 CH1 online map
SW116 ~ SW119	TOSLINE-S20 CH2 online map
SW120 ~ SW123	TOSLINE-S20 CH1 standby map
SW124 ~ SW127	TOSLINE-S20 CH2 standby map
SW128 ~ SW191	TOSLINE-S20 scan healthy map
SW192 ~ SW255	Reserve (for future use)

Special device	Name	Function		
S0000	Operation mode	0: Initializing	4: HOLD mode	B: D-STOP
S0001		1: HALT mode	6: ERROR mode	D: S-HALT
S0002		2: RUN mode	9: D-HALT	E: S-RUN
S0003		3: Run-F mode	A: D-RUN	F: S-STOP
S0004	CPU error (Down)	ON when error occurs (OR condition of related flag in SW001)		
S0005	I/O error (Down)	ON when error occurs (OR condition of related flag in SW002)		
S0006	Program error (Down)	ON when error occurs (OR condition of related flag in SW003)		
S0007	EEPROM alarm (Warning)	ON when EEPROM number of writing times 100,000 exceeded (operation continues)		
S0008	Constant scan delay (Warning)	ON when actual scan time exceeds the constant scan time setting		
S0009	I/O alarm (Warning)	ON when I/O error detected by I/O error mapping		
S000A	Calendar LSI error (Warning)	ON when clock-calendar data fault (operation continues)		
S000B		Reserve (for futuer use)		
S000C				
S000D	TOSLINE-F10 error (Warning)	ON when TOSLINE-F10 error (operation continues)		
S000E	TOSLINE-S20 error (Warning)	ON when TOSLINE-S20 error (operation continues)		
S000F	Battery voltage low (Warning)	ON when battery voltage low (operation continues)		
S0010	System ROM error (Down)	ON when system ROM error		
S0011	Sytemr RAM error (Down)	ON when system RAM error		
S0012	Program memory error (Down)	ON when program memory (RAM) error		
S0013	EEPROM error (Down)	ON when EEPROM error		
S0014	IC memory card error (Down)	ON when IC memory card error		
S0015	LP error (Down)	ON when language processor (LP) error		
S0016	Main CPU error (Down)	ON when main CPU error		
S0017		Reserve (for future use)		
S0018				
S0019				
S001A				
S001B				
S001C	Power interrupution recovery			
S001D	Power interrupution detect			
S001E		Reserve (for future use)		
S001F	Watch-dog timer error (Down)	ON when watch-dog timer error occurs		

*1) This area is for reference only (Do not write)
*2) The error flags are reset at the beginning of RUN mode.

Special device	Name	Function
S0020	I/O bus error (Down)	ON when I/O bus error occurs
S0021	I/O mismatch error (Down)	ON when I/O mismatch error occurs (allocation infomation and mounting state do not agree)
S0022	I/O response error (Down)	ON when no I/O response occurs
S0023	I/O parity error (Down)	ON when I/O data parity error occurs.
S0024	Expansion power error (Down)	ON when expansion unit power error occurs
S0025	I/O interrupt error (Warning)	ON when unused I/O interrupt occurs (operation continues)
S0026	Special module error (Warning)	ON when fault occurs in special module (operation continues)
S0027		Reserve(for future use)
S0028		
S0029		
S002A		
S002B		
S002C		
S002D		
S002E		
S002F		
S0030	Program error	ON when program error occurs (OR condition of SW006 flags)
S0031	Scan time error (Down)	ON when scan cycle exceeds the limit value
S0032		Reserve (for future use)
S0033		
S0034		
S0035		
S0036		
S0037		
S0038		
S0039		
S003A		
S003B		
S003C		
S003D	IC memory card mounting status	ON when IC memory card mounted
S003E	IC memory card write protect	ON when in write protect state
S003F	IC memory card battery low (Warning)	ON when voltage drop of battery housed in IC memory card.

*1) This area is for reference only (Do not write)
*2) The error flags are reset at the beginning of RUN mode.

Special device	Name	Function	
S0040	Timing relay 0.1sec	0.05sec OFF/0.05sec ON (Cycle 0.1sec)	All OFF when RUN starts up
S0041	Timing relay 0.2sec	0.1sec OFF/0.1sec ON (Cycle 0.2sec)	
S0042	Timing relay 0.4sec	0.2sec OFF/0.2sec ON (Cycle 0.4sec)	
S0043	Timing relay 0.8sec	0.4sec OFF/0.4sec ON (Cycle 0.8sec)	
S0044	Timing relay 1.0sec	0.5sec OFF/0.5sec ON (Cycle 1.0sec)	
S0045	Timing relay 2.0sec	1.0sec OFF/1.0sec ON (Cycle 2.0sec)	
S0046	Timing relay 4.0sec	2.0sec OFF/2.0sec ON (Cycle 4.0sec)	
S0047	Timing relay 8.0sec	4.0sec OFF/4.0sec ON (Cycle 8.0sec)	
S0048		Reserve (for future use)	
S0049			
S004A			
S004B			
S004C			
S004D			
S004E	Always OFF	Always OFF	
S004F	Always ON	Always ON	
S0050	CF (carry flag)	Used by instructions with carry	
S0051	ERF (Error flag)	ON through error occurrence when executing instructions (linked with each error flag of SW006)	
S0052		Reserve (for future use)	
S0053			
S0054			
S0055			
S0056			
S0057			
S0058			
S0059			
S005A			
S005B			
S005C			
S005D			
S005E			
S005F			

*) This area (except for S0050, S0051) is for reference only (writing is ineffective)

Special device	Name	Function
S0060	Illegal instruction detection (Down)	ON when illegal instruction detected
S0061		Reserve (for future use)
S0062		
S0063		
S0064	Boundary error (Warning)	ON when address range exceeded by indirect address designation (operation continues)
S0065	Address boundary error (Warning)	ON when destination (indirect) error by CALL instruction or JUMP instruction (operation continues)
S0066	Program memory parity error (Down)	ON when parity error occurs in user program memory
S0067	Data memory parity error (Down)	ON when parity error occurs in user data memory
S0068	Division error (Warning)	ON when error occurs by division instruction (operation continues)
S0069	BCD data error (Warning)	ON when fault data detected by BCD instruction (operation continues)
S006A	Table operation error (Warning)	ON when table limits exceeded by table operation instruction (operation continues)
S006B	Encode error (Warning)	ON when error occurs by encode instruction (operation continues)
S006C	Address registration error (Warning)	ON when destination for CALL instruction or JUMP instruction unregistered (operation continues)
S006D	Nesting error (Warning)	ON when nesting exceeded by CALL instruction, FOR instruction or MCSn instruction (operation continues)
S006E		Reserve (for future use)
S006F		

*1) The error flags are reset at the beginning of RUN mode.
*2) For warning flags, resetting by user program is possible.

Special register	Name	Function	
SW007	Calendar data (Year)	Last 2 digits of the calendar year (91, 92, ...)	Stored in lower 8 bits by BCD code
SW008	Calendar data (Month)	Month (01-12)	
SW009	Calendar data (Day)	Day (01-31)	
SW010	Calendar data (Hour)	Hour (00-23)	
SW011	Calendar data (Minute)	Minute (00-59)	
SW012	Calendar data (Second)	Second (00-59)	
SW013	Calendar data (Day of the week)	Day of the week (Sunday=00, Monday=01, ...Saturday=06)	

*1) The clock-calendar data setting is performed by calendar setting instruction (CLND) or by calendar setting operation by programmer. (It is ineffective to write data directly to the special registers)
*2) When the data cannot be read correctly due to the calendar LSI fault, these registers become H00FF.
*3) Calendar accuracy is ± 30 seconds/month.

Special device	Name	Function
S0140	Bit/register check	Bit pattern/register value check is executed by setting ON
S0141	Bit/register check result	ON when either S0142-S0146 is ON
S0142	Bit pattern check result	ON when bit pattern check error detected
S0143	Register value check result(1)	ON when register value check error detected for register 1
S0144	Register value check result(2)	ON when register value check error detected for register 2
S0145	Register value check result(3)	ON when register value check error detected for register 3
S0146	Register value check result(4)	ON when register value check error detected for register 4
S0147		Reserve (for future use)
S0148		
S0149		
S014A		
S014B		
S014C		
S014D		
S014E		
S014F		
S0150	I/O error mapping	I/O error mapping is executed by setting ON
S0151		Reserve (for future use)
S0152		
S0153		
S0154		
S0155		
S0156		
S0157		
S0158		
S0159		
S015A		
S015B		
S015C		
S015D		
S015E		
S015F	Checkpoint for bit/register check	OFF: before program execution ON: after program execution

Special register	Name	Function
SW016	First error code	• The designated error codes (1-64) are stored in order of execution in SW018-SW033 (the earlier the code, the lower the address), and the number of registration (SW017) is updated. • The earliest error code occurring (the content of SW018) is stored in the leading error code (SW016). • The registered error codes are cancelled one by one by the execution of the diagnostic display reset instruction or by a reset operation by the programmer. At this time, the number of registers is reduced by 1 and the storage positions of the error codes are shifted up.
SW017	Number of registration	
SW018	Error code (First)	
SW019	Error code (2)	
SW020	Error code (3)	
SW021	Error code (4)	
SW022	Error code (5)	
SW023	Error code (6)	
SW024	Error code (7)	
SW025	Error code (8)	
SW026	Error code (9)	
SW027	Error code (10)	
SW028	Error code (11)	
SW029	Error code (12)	
SW030	Error code (13)	
SW031	Error code (14)	
SW032	Error code (15)	
SW033	Error code (16)	

Special device	Name	Function
S0340	Annunciator relay 1	• The annunciator relays corresponding to the error codes registered in SW018-SW033 become ON
S0341	Annunciator relay 2	
S0342	Annunciator relay 3	
S0343	Annunciator relay 4	
S0344	Annunciator relay 5	
S0345	Annunciator relay 6	
S0346	Annunciator relay 7	
S0347	Annunciator relay 8	
S0348	Annunciator relay 9	
S0349	Annunciator relay 10	
S034A	Annunciator relay 11	
S034B	Annunciator relay 12	
S034C	Annunciator relay 13	
S034D	Annunciator relay 14	
S034E	Annunciator relay 15	
S034F	Annunciator relay 16	

Special device	Name	Function
S0350	Annunciator relay 17	• The annunciator relays corresponding to the error codes registered in SW018-SW033 become ON
S0351	Annunciator relay 18	
S0352	Annunciator relay 19	
S0353	Annunciator relay 20	
S0354	Annunciator relay 21	
S0355	Annunciator relay 22	
S0356	Annunciator relay 23	
S0357	Annunciator relay 24	
S0358	Annunciator relay 25	
S0359	Annunciator relay 26	
S035A	Annunciator relay 27	
S035B	Annunciator relay 28	
S035C	Annunciator relay 29	
S035D	Annunciator relay 30	
S035E	Annunciator relay 31	
S035F	Annunciator relay 32	
S0360	Annunciator relay 33	
S0361	Annunciator relay 34	
S0362	Annunciator relay 35	
S0363	Annunciator relay 36	
S0364	Annunciator relay 37	
S0365	Annunciator relay 38	
S0366	Annunciator relay 39	
S0367	Annunciator relay 40	
S0368	Annunciator relay 41	
S0369	Annunciator relay 42	
S036A	Annunciator relay 43	
S036B	Annunciator relay 44	
S036C	Annunciator relay 45	
S036D	Annunciator relay 46	
S036E	Annunciator relay 47	
S036F	Annunciator relay 48	

Special device	Name	Function
S0370	Annunciator relay 49	• The annunciator relays corresponding to the error codes registered in SW018-SW033 become ON
S0371	Annunciator relay 50	
S0372	Annunciator relay 51	
S0373	Annunciator relay 52	
S0374	Annunciator relay 53	
S0375	Annunciator relay 54	
S0376	Annunciator relay 55	
S0377	Annunciator relay 56	
S0378	Annunciator relay 57	
S0379	Annunciator relay 58	
S037A	Annunciator relay 59	
S037B	Annunciator relay 60	
S037C	Annunciator relay 61	
S037D	Annunciator relay 62	
S037E	Annunciator relay 63	
S037F	Annunciator relay 64	

Special device	Name	Function
S0390	Timer interrupt execution status	ON during execution
S0391	I/O interrupt #1 execution status	ON during execution
S0392	I/O interrupt #2 execution status	ON during execution
S0393	I/O interrupt #3 execution status	ON during execution
S0394	I/O interrupt #4 execution status	ON during execution
S0395	I/O interrupt #5 execution status	ON during execution
S0396	I/O interrupt #6 execution status	ON during execution
S0397	I/O interrupt #7 execution status	ON during execution
S0398	I/O interrupt #8 execution status	ON during execution
S0399		Reserve (for future use)
S039A		
S039B		
S039C		
S039D		
S039E		
S036F		

Special device	Name	Function
S0400	Hot restart mode	ON when hot restart mode (setting by program is available)
S0401	HOLD device	ON during HOLD mode (setting by program is available)
S0402		Reserve (for future use)
S0403	Sub-program #2 mode	Sub-program #2 mode setting (OFF: Normal ON: Special)
S0404		Reserve (for future use)
S0405	Sub-program #2 execution mode	Sub-program #2 execution mode setting (OFF: One time ON: Cyclic)
S0406	Sub-program #3 execution mode	Sub-program #3 execution mode setting (OFF: One time ON: Cyclic)
S0407	Sub-program #4 execution mode	Sub-program #4 execution mode setting (OFF: One time ON: Cyclic)
S0408		Reserve (for future use)
S0409	Sub-program #2 request	Sub-program #2 request command (Execution request by setting ON)
S040A	Sub-program #3 request	Sub-program #3 request command (Execution request by setting ON)
S040B	Sub-program #4 request	Sub-program #4 request command (Execution request by setting ON)
S040C S040D S040E S040F		Reserve (for future use)
S0410	Sub-program #1 execution status	ON during sub-program #1 execution
S0411	Sub-program #2 execution status	ON during sub-program #2 execution
S0412	Sub-program #3 execution status	ON during sub-program #3 execution
S0413	Sub-program #4 execution status	ON during sub-program #4 execution
S0414		Reserve (for future use)
S0415	Sub-program #2 delay (Warning)	ON when sub-program #2 execution delay (cyclic mode)
S0416	Sub-program #3 delay (Warning)	ON when sub-program #3 execution delay (cyclic mode)
S0417	Sub-program #4 delay (Warning)	ON when sub-program #4 execution delay (cyclic mode)
S0418 S0419 S041A S041B S041C S041D S041E S041F		Reserve (for future use)

Special register	Name	Function
SW042	Sub-program #2 interval	Number of scans for sub-program #2 cyclic mode
SW043	Sub-program #3 interval	Number of scans for sub-program #3 cyclic mode
SW044	Sub-program #4 interval	Number of scans for sub-program #4 cyclic mode
SW045	Continuous operation time	Continuous operation time setting for power interruption shut down

Special device	Name	Function
S0460	I/O error map #0-0	ON when I/O error detected in unit 0-slot 0
S0461	I/O error map #0-1	ON when I/O error detected in unit 0-slot 1
S0462	I/O error map #0-2	ON when I/O error detected in unit 0-slot 2
S0463	I/O error map #0-3	ON when I/O error detected in unit 0-slot 3
S0464	I/O error map #0-4	ON when I/O error detected in unit 0-slot 4
S0465	I/O error map #0-5	ON when I/O error detected in unit 0-slot 5
S0466	I/O error map #0-6	ON when I/O error detected in unit 0-slot 6
S0467	I/O error map #0-7	ON when I/O error detected in unit 0-slot 7
S0468	I/O error map #0-8	ON when I/O error detected in unit 0-slot 8
S0469	I/O error map #0-9	ON when I/O error detected in unit 0-slot 9
S046A		Reserve (for future use)
S046B		
S046C		
S046D		
S046E		
S046F		
S0470	I/O error map #1-0	ON when I/O error detected in unit 1-slot 0
S0471	I/O error map #1-1	ON when I/O error detected in unit 1-slot 1
S0472	I/O error map #1-2	ON when I/O error detected in unit 1-slot 2
S0473	I/O error map #1-3	ON when I/O error detected in unit 1-slot 3
S0474	I/O error map #1-4	ON when I/O error detected in unit 1-slot 4
S0475	I/O error map #1-5	ON when I/O error detected in unit 1-slot 5
S0476	I/O error map #1-6	ON when I/O error detected in unit 1-slot 6
S0477	I/O error map #1-7	ON when I/O error detected in unit 1-slot 7
S0478	I/O error map #1-8	ON when I/O error detected in unit 1-slot 8
S0479	I/O error map #1-9	ON when I/O error detected in unit 1-slot 9
S047A	I/O error map #1-10	ON when I/O error detected in unit 1-slot 10
S047B		Reserve (for future use)
S047C		
S047D		
S047E		
S047F		

Special device	Name	Function
S0480	I/O error map #2-0	ON when I/O error detected in unit 2 - slot 0
S0481	I/O error map #2-1	ON when I/O error detected in unit 2 - slot 1
S0482	I/O error map #2-2	ON when I/O error detected in unit 2 - slot 2
S0483	I/O error map #2-3	ON when I/O error detected in unit 2 - slot 3
S0484	I/O error map #2-4	ON when I/O error detected in unit 2 - slot 4
S0485	I/O error map #2-5	ON when I/O error detected in unit 2 - slot 5
S0486	I/O error map #2-6	ON when I/O error detected in unit 2 - slot 6
S0487	I/O error map #2-7	ON when I/O error detected in unit 2 - slot 7
S0488	I/O error map #2-8	ON when I/O error detected in unit 2 - slot 8
S0489	I/O error map #2-9	ON when I/O error detected in unit 2 - slot 9
S048A	I/O error map #2-10	ON when I/O error detected in unit 2 - slot 10
S048B		Reserve (for future use)
S048C		
S048D		
S048E		
S048F		
S0490	I/O error map #3-0	ON when I/O error detected in unit 3 - slot 0
S0491	I/O error map #3-1	ON when I/O error detected in unit 3 - slot 1
S0492	I/O error map #3-2	ON when I/O error detected in unit 3 - slot 2
S0493	I/O error map #3-3	ON when I/O error detected in unit 3 - slot 3
S0494	I/O error map #3-4	ON when I/O error detected in unit 3 - slot 4
S0495	I/O error map #3-5	ON when I/O error detected in unit 3 - slot 5
S0496	I/O error map #3-6	ON when I/O error detected in unit 3 - slot 6
S0497	I/O error map #3-7	ON when I/O error detected in unit 3 - slot 7
S0498	I/O error map #3-8	ON when I/O error detected in unit 3 - slot 8
S0499	I/O error map #3-9	ON when I/O error detected in unit 3 - slot 9
S049A	I/O error map #3-10	ON when I/O error detected in unit 3 - slot 10
S049B		Reserve (for future use)
S049C		
S049D		
S049E		
S049F		

Special device	Name		Function
S0780	TOSLINE-F10 CH1 command	Transmission status	ON during transmission
S0781		Output inhibit status	ON when output inhibit mode
S0782		Re-configuration	ON during re-configuration
S0783			Reserve (for future use)
S0784		Scan transmission error	ON when scan transmission error occurs
S0785			Reserve (for future use)
S0786			
S0787			
S0788		Transmission stop	Transmission stop by setting ON
S0789		Output inhibit	Output inhibit by setting ON
S078A			Reserve (for future use)
S078B			
S078C			
S078D			
S078E			
S078F			
S0790	TOSLINE-F10 CH1 status	Transmission status	ON during transmission
S0791		Scan transmission	ON during scan transmission
S0792			Reserve (for future use)
S0793			
S0794			
S0795		MS operation mode	OFF: Normal mode ON: Test mode
S0796			Reserve (for future use)
S0797			
S0798			
S0799			
S079A			
S079B			
S079C			
S079D			
S079E			
S079F			

*) Refer to the TOSLINE-F10 manual for details.

Special register	Name	Function
SW080	TOSLINE-F10 CH2 command	• Bit assignment in the register is the same as SW078 and SW079.
SW081	TOSLINE-F10 CH2 status	
SW082	TOSLINE-F10 CH3 cammand	
SW083	TOSLINE-F10 CH3 status	
SW084	TOSLINE-F10 CH4 command	
SW085	TOSLINE-F10 CH4 status	
SW086	TOSLINE-F10 CH5 command	
SW087	TOSLINE-F10 CH5 status	
SW088	TOSLINE-F10 CH6 command	
SW089	TOSLINE-F10 CH6 status	
SW090	TOSLINE-F10 CH7 command	
SW091	TOSLINE-F10 CH7 status	
SW092	TOSLINE-F10 CH8 command	
SW093	TOSLINE-F10 CH8 status	

Special register	Name		Function
SW094	TOSLINE-F10 scan error map	LW000~LW015	• The corresponding bit comes ON when the LW register is not updated normally. • The lowest address of LW register corresponds to bit 0 in the SW register, and in the order.
SW095		LW016~LW031	
SW096		LW032~LW047	
SW097		LW048~LW063	
SW098		LW064~LW079	
SW099		LW080~LW095	
SW100		LW096~LW111	
SW101		LW112~LW127	
SW102		LW128~LW143	
SW103		LW144~LW159	
SW104		LW160~LW175	
SW105		LW176~LW191	
SW106		LW192~LW207	
SW107		LW208~LW223	
SW108		LW224~LW239	
SW109		LW240~LW255	

Special device	Name		Function
S1100	TOSLINE-S20 CH1 station status	Test mode	ON when test mode
S1101			Reserve(for future use)
S1102			
S1103			
S1104		Master/slave	ON when master station
S1105		Scan inhibit	ON when scan transmission inhibited
S1106			Reserve(for future use)
S1107			
S1108			
S1109			
S110A			
S110B			
S110C		Online	ON when online mode
S110D		Standby	ON when standby mode
S110E		Offline	ON when offline mode
S110F		Down	ON when down mode
S1110	TOSLINE-S20 CH2 station status	Test mode	ON when test mode
S1111			Reserve(for future use)
S1112			
S1113			
S1114		Master/slave	ON when master station
S1115		Scan inhibit	ON when scan transmission inhibited
S1116			Reserve(for future use)
S1117			
S1118			
S1119			
S111A			
S111B			
S111C		Online	ON when online mode
S111D		Standby	ON when standby mode
S111E		Offline	ON when offline mode
S111F		Down	ON when down mode

*) Refer to the TOSLINE-S20 manual for details.

Special register	Name		Function
SW112	TOSLINE-S20 CH1 online map	station No.1~No.16	• The corresponding bit is ON when the station is online. • The lowest station number corresponds to bit 0 in the SW register, and in the order.
SW113		station No.17~No.32	
SW114		station No.33~No.48	
SW115		station No.49~No.64	
SW116	TOSLINE-S20 CH2 online map	station No.1~No.16	
SW117		station No.17~No.32	
SW118		station No.33~No.48	
SW119		station No.49~No.64	
SW120	TOSLINE-S20 CH1 standby map	station No.1~No.16	• The corresponding bit is ON when the station is standby. • The lowest station number corresponds to bit 0 in the SW register, and in the order.
SW121		station No.17~No.32	
SW122		station No.33~No.48	
SW123		station No.49~No.64	
SW124	TOSLINE-S20 CH2 standby map	station No.1~No.16	
SW125		station No.17~No.32	
SW126		station No.33~No.48	
SW127		station No.49~No.64	

Special register	Name		Function
SW128	TOSLINE-S20 scan healthy map	W0000~W0015	• The corresponding bit is ON when the W register is updated normally. • The lowest address of W register corresponds to bit 0 in the SW register, and in the order.
SW129		W0016~W0031	
SW130		W0032~W0047	
SW131		W0048~W0063	
SW132		W0064~W0079	
SW133		W0080~W0095	
SW134		W0096~W0111	
SW135		W0112~w0127	
SW136		W0128~W0143	
SW137		W0144~W0159	
SW138		W0160~W0175	
SW139		W0176~W0191	
SW140		W0192~W0207	
SW141		W0208~W0223	
SW142		W0224~W0239	
SW143		W0240~W0255	

Special register	Name		Function
SW144	TOSLINE-S20 scan healthy map	W0256~W0271	• The corresponding bit is ON when the W register is updated normally. • The lowest address of W register corresponds to bit 0 in the SW register, and in the order.
SW145		W0272~W0278	
SW146		W0288~W0303	
SW147		W0304~W0319	
SW148		W0320~W0335	
SW149		W0336~W0351	
SW150		W0352~W0367	
SW151		W0368~W0383	
SW152		W0384~W0399	
SW153		W0400~W0415	
SW154		W0416~W0431	
SW155		W0432~W0447	
SW156		W0448~W0463	
SW157		W0464~W0479	
SW158		W0480~W0495	
SW159		W0496~W0511	
SW160		W0512~W0527	
SW161		W0528~W0543	
SW162		W0544~W0559	
SW163		W0560~W0575	
SW164		W0576~W0591	
SW165		W0592~W0607	
SW166		W0608~W0623	
SW167		W0624~W0639	
SW168		W0640~W0655	
SW169		W0656~W0671	
SW170		W0672~W0687	
SW171		W0688~W0703	
SW172		W0704~W0719	
SW173		W0720~W0735	
SW174		W0736~W0751	
SW175		W0752~W0767	

Special register	Name		Function
SW176	TOSLINE-S20 scan healthy map	W0768~W0783	• The corresponding bit is ON when the W register is updated normally. • The lowest address of W register corresponds to bit 0 in the SW register, and in the order.
SW177		W0784~W0799	
SW178		W0800~W0815	
SW179		W0816~W0831	
SW180		W0832~W0847	
SW181		W0848~W0863	
SW182		W0864~W0879	
SW183		W0880~W0895	
SW184		W0896~W0911	
SW185		W0912~W0927	
SW186		W0928~W0943	
SW187		W0944~W0959	
SW188		W0960~W0975	
SW189		W0976~W0991	
SW190		W0992~W1007	
SW191		W1008~W1023	

3.3 Register data types

It has already been explained the register is "a location which stores 16 bits of data". In the T3 instructions, the following types of data can be processed using single registers or multiple consecutive registers.

- Unsigned integers (integers in the range 0 to 65535)
- Integers (integers in the range -32768 to 32767)
- BCD (integers in the range 0 to 9999 expressed by BCD code)
- Unsigned double-length integers (integers in the range 0 to 4294967295)
- Double-length integers (integers in the range -2147483648 to 2147483647)
- Double-length BCD (integers in the range 0 to 99999999 expressed by BCD code)
- Floating point data (real number in the range -3.40282×10^{38} to 3.40282×10^{38})

However, there are no dedicated registers corresponding to the types for processing these types of data. The processing of the register data varies according to which instruction is used.

In other words, as shown in the following example, even when the same register is used, if the data type of the instruction differs, the processing of the register data will also differ.

Example)

When the value of D0005 is HFFFF (hexadecimal FFFF):

(1) In the unsigned comparison instruction (Greater than),

$\dashv[$ D0005 U $>$ 100 $]\vdash$ decision output (ON when true)

The value of D0005 is regarded as 65535 (unsigned integer), therefore it is judged to be greater than the compared value (100) and the output of the instruction becomes ON.

(2) In the (signed) comparison instruction (Greater than),

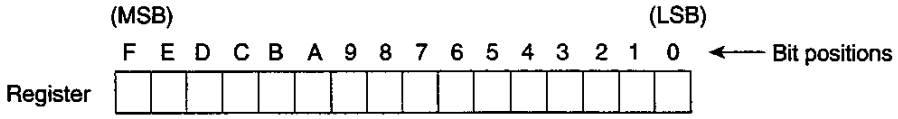
$\dashv[$ D0005 $>$ 100 $]\vdash$ decision output (ON when true)

The value of D0005 is regarded as -1 (integer), therefore it is judged not to be greater than the compared value (100) and the output of the instruction becomes OFF.

In this way, since there is no classification of registers by data type, it is possible to execute complex data operations provided their use is thoroughly understood. However, in order to make the program easier to see, it is recommended that registers be used by allocation by data types (1 register is processed by 1 data type) as far as possible.

(1) Unsigned Integer

This is a 16-bit unsigned integer expressed by 1 register. The bit configuration inside the register is as shown below.



Bit 0 is the least significant bit (LSB), and bit F is the most significant bit (MSB). The processable numerical value range is as shown in the following Table.

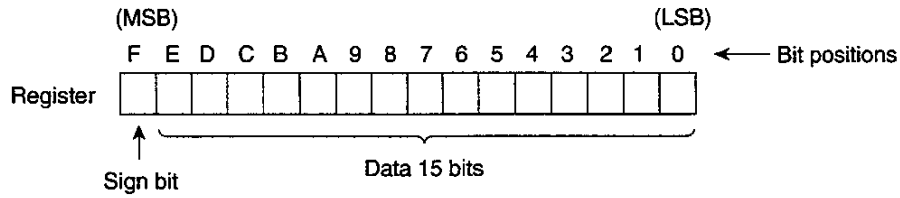
Numerical Value (Decimal)	Binary Expression	Hexadecimal Expression
65535	1111 1111 1111 1111	FFFF
65534	1111 1111 1111 1110	FFFE
∫	∫	∫
1	0000 0000 0000 0001	0001
0	0000 0000 0000 0000	0000

NOTE

When programming and when program monitoring, it is possible to change between decimal numbers and hexadecimal numbers for displaying/setting register data. When using a hexadecimal display, "H" is attached before the numerical value.
Example) H89AB (hexadecimal 89AB)

(2) Integer

This is a 16-bit integer expressed by 1 register. A negative number is expressed by 2's complement.



The numerical value is expressed by the 15 bits from bit 0 to bit E. Bit F expresses the sign (0 when positive, 1 when negative)

Processable numerical range and expression format are shown in the following Table.

Numerical Value (Decimal)	Binary Expression	Hexadecimal Expression
32767	0111 1111 1111 1111	7FFF
32766	0111 1111 1111 1110	7FFE
∫	∫	∫
1	0000 0000 0000 0001	0001
0	0000 0000 0000 0000	0001
-1	1111 1111 1111 1111	FFFF
∫	∫	∫
-32767	1000 0000 0000 0001	8001
-32768	1000 0000 0000 0000	8000

The 2's complement is that the lower 16 bits become all 0 by adding the 2's complement data and the original data.

Example)

```
      0111 1111 1111 1111   (Binary)=32767
+     1000 0000 0000 0001   (Binary)=-32767
  1 0000 0000 0000 0000
```

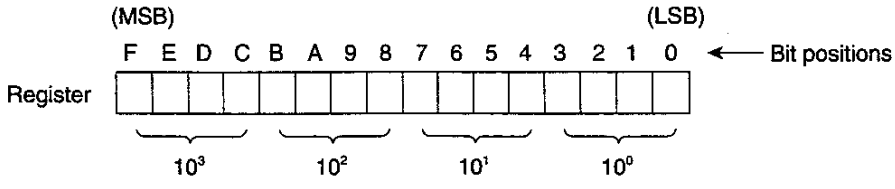
In calculation, the 2's complements of a numerical value can be found by the operation of inverting each bit of that numerical value and adding 1.

Example)

```
0111 1111 1111 1111   (Binary)=32767
   (bit inversion)
1000 0000 0000 0000   (Binary)=-32768
     (add 1)
1000 0000 0000 0001   (Binary)=-32767
```

(3) BCD

BCD is the abbreviation of Binary Coded Decimal. BCD expresses 1 digit (0-9) of a decimal number by 4 bits of a binary number. Therefore, 1 register can express the numerical value of a 4-digit decimal number.



Processable numerical range and expression format are shown in the following Table.

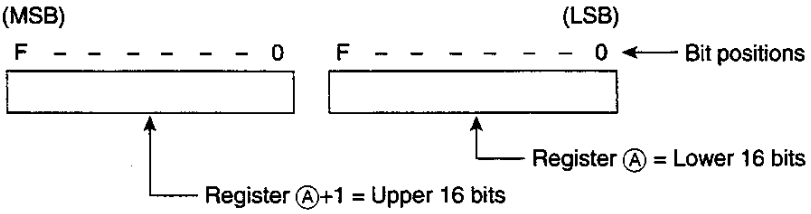
Numerical Value (Decimal)	Binary Expression	Hexadecimal Expression
9999	1001 1001 1001 1001	9999
9998	1001 1001 1001 1000	9998
∫	∫	∫
10	0000 0000 0001 0000	0010
9	0000 0000 0000 1001	0009
∫	∫	∫
1	0000 0000 0000 0001	0001
0	0000 0000 0000 0000	0000

NOTE

Basically, BCD is a data format used for data inputs from BCD-output type numerical setting devices and data outputs to BCD-input type numerical display devices. However, the T3 is provided with dedicated instructions which execute the calculations on BCD data as they stand.

(4) Unsigned Double-Length Integer

This is 32-bit unsigned integer which is expressed using 2 consecutive registers. In the case of double-length data, the registers are designated in the form Ⓐ+1 •Ⓐ. Ⓐ indicates the lower 16 bits and Ⓐ+1 shows the upper 16 bits. (Ⓐ+1 is the register following register Ⓐ)



Example) When processing an unsigned double-length integer in double length register D0201•D0200, D0200 becomes Ⓐ and D0201 becomes Ⓐ+1. D0200 becomes the lower side and D0201 becomes the upper side.
In programming, when D0200 is entered in the position which designates the double-length operand, D0201•D0200 is automatically displayed.

The numerical value range in which unsigned double-length integers can be processed is shown in the table on the following page.

Numerical Value	Hexadecimal Expression	
	Register Ⓐ+1	Register Ⓐ
4294967295	FFFF	FFFF
∫	∫	∫
65536	0001	0000
65535	0000	FFFF
∫	∫	∫
0	0000	0000

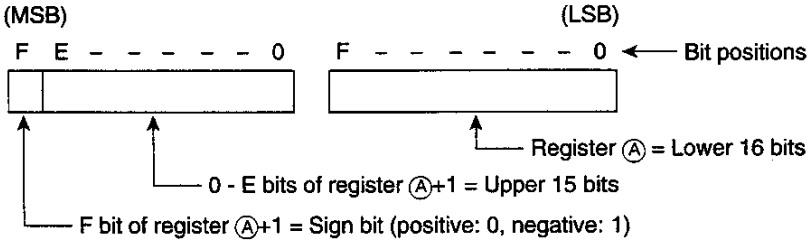
NOTE

Both odd-numbered addresses and even-numbered addresses may be used as register Ⓐ.

(5) Double-Length Integer

This is 32-bit integer which is expressed using 2 consecutive registers. Negative numbers are expressed by 2's complement. (See (2) 'Integers')

The registers are designated in the form Ⓐ+1 • Ⓐ. Ⓐ becomes the lower and Ⓐ+1 becomes the upper.


(MSB)
(LSB)
F E - - - - - - 0
F - - - - - - - 0
Bit positions
Register Ⓐ = Lower 16 bits
0 - E bits of register Ⓐ+1 = Upper 15 bits
F bit of register Ⓐ+1 = Sign bit (positive: 0, negative: 1)

The numerical value is expressed by the 31 bits from bit 0 of register Ⓐ to bit E of register Ⓐ+1. The sign is expressed by bit F of register Ⓐ+1 (0 when positive, 1 when negative).

Example) When a double-length integer is processed by registers D1002•D1001, D1001 becomes Ⓐ and D1002 becomes Ⓐ+1, and D1001 is the lower and D1002 is the upper. Also, the sign is expressed by the bit F of D1002.

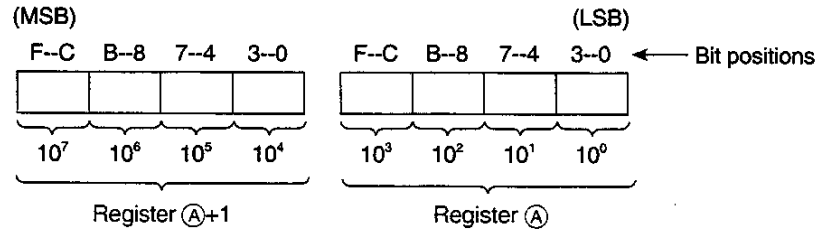
In programming, when D1001 is entered in the position which designates the double-length operand, D1002•D1001 is automatically displayed.

The numerical value range in which double-length integers can be processed is shown in the table on the following page.

Numerical Value	Hexadecimal Expression	
	Register Ⓐ+1	Register Ⓐ
2147483647	7FFF	FFFF
∫	∫	∫
65536	0001	0000
65535	0000	FFFF
∫	∫	∫
0	0000	0000
-1	FFFF	FFFF
∫	∫	∫
-65536	FFFF	0000
-65537	FFFE	FFFF
∫	∫	∫
-2147483648	8000	0000

(6) Double-Length BCD

This is 8-digit BCD data which is expressed by using 2 consecutive registers.



The registers are designated in the form Ⓐ+1•Ⓐ, and Ⓐ becomes the lower 4 digits while Ⓐ+1 becomes the upper 4 digits.

Example) When processing a double-length BCD by registers XW001 •XW000, XW000 becomes Ⓐ while XW001 becomes Ⓐ+1 and XW000 becomes the lower 4 digits while XW001 becomes the upper 4 digits.

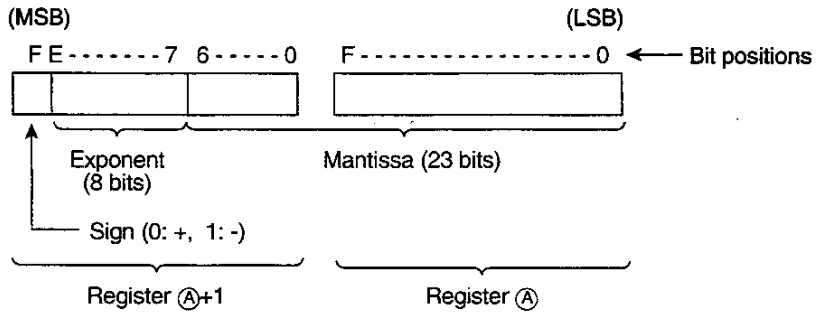
The following table shows the numerical range and the expression format in which double-length BCD data can be processed.

Numerical Value	Hexadecimal Expression	
	Register Ⓐ+1	Register Ⓐ
99999999	9999	9999
∫	∫	∫
1	0000	0001
0	0000	0000

(7) Floating Point Data

This is a real number which is expressed using 2 consecutive registers (32-bit).

The registers are designated in the form Ⓐ+1•Ⓐ. Internally, the following format is used. (conforms to IEEE754)



Value = (Sign)1.(Mantissa) × $2^{(Exponent-127)}$

The floating point data is used with the following floating point instructions. Therefore, there is no need for user to consider the format.

- Conversions (Floating point ↔ Double-length integer)
- Floating point arithmetics
- Floating point comparisons
- Floating point functions (Trigonometrics, square root, etc.)
- Floating point process operations (Integral, PID, etc.)

The following table shows the numerical range in which the floating point data can be processed.

Numerical value	Expression	Remarks
3.40282×10^{38}	3.40282E38	Maximum
∫	∫	
1.17549×10^{-38}	1.17549E-38	Nearest to 0
0	0	
-1.17549×10^{-38}	-1.17549E-38	Nearest to 0
∫	∫	
-3.40282×10^{38}	-3.40282E38	Minimum

3.4 Index modification

When registers are used by instructions, the method of directly designating the register address as shown in Example 1) below is called 'direct addressing'.

As opposed to this, the method of indirectly designating the register by combination with the contents of the index registers (I, J, K) as shown in Example 2) below is called the 'indirect addressing'. In particular, in this case, since the address is modified using an index register, this is called 'index modification'.

Example 1)

```
-[ RW 100 MOV D3500 ]-
```

Data transfer instruction
Transfer content of RW100 to D3500

Example 2)

```
          I         J
-[ RW 100 MOV D3500 ]-
```

Data transfer instruction (index modification attached)
Transfer content of RW(100+I) to D(3500+J)
(If I=3 and J=200, the content of RW 103 is transferred to D3700)

There are 3 types of index register, I, J and K. Each type processes 16-bit integers (-32768 to 32767). There are no particular differences in function between these 3 types of index register.

There is no special instruction for substituting values in these index registers. There are designated as destination for normal instructions.

Example 1) Substituting a constant in an index register

```
-[ 64 MOV I ]-   (Substitute 64 in index register I)

-[ -2 MOV J ]-   (Substitute -2 in index register J)
```

Example 2) Substituting register data in an index register

```
-[ D0035 MOV K ]-   (Substitute the value of D0035 in index
                     register K)
-[ RW078 MOV I ]-   (Substitute the value of RW078 in index
                     register I)
```

Example 3) Substituting the result of an operation in an index register

-[RW200 - 30 → I]-

(Substitute the result of subtracting 30 from RW200 in I)

-[XW004 ENC (4) J]-

(Substitute the uppermost ON bit position of XW004 in J (encode))

NOTE

Although, basically, index registers are processed as single-length (16 bits), when, for instance, using an index register as the storage destination for a instruction which becomes double-length as the result of a multiplication instruction or the like, only the combinations J • I or K • J are effective. In this case, it becomes J • I by designating I in the double-length operand position, and J becomes upper while I becomes lower. In the same, by designating J, it becomes K • J, and K becomes upper while J becomes lower.
Example)

-[D1357 * 10 → J • I]-

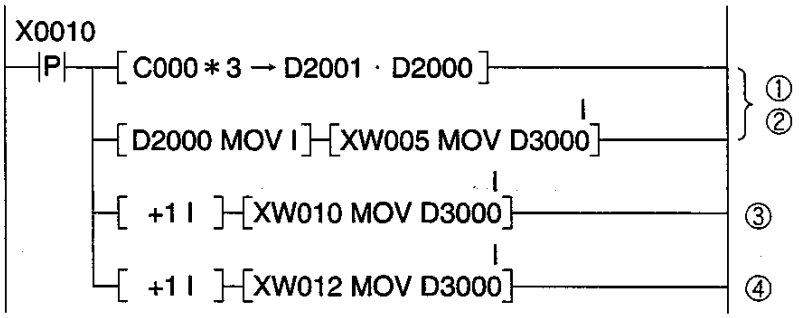
The following are examples of registers in which index modification has been executed.

I RW100	When I = 0,	expresses RW100
	When I = 1,	expresses RW101
	When I = -1,	expresses RW099
	When I = 100,	expresses RW200
	When I = -100,	expresses RW000

J D0201 · D0200	When J = 0,	expresses D0201 · D0200
	When J = 1,	expresses D0202 · D0201
	When J = 2,	expresses D0203 · D0202
	When J = -1,	expresses D0200 · D0199
	When J = -2,	expresses D0199 · D0198

The following shows an example of the operation when index modification is applied to a program.

Example)



The following processing is carried out when X0010 changes from OFF to ON.

① Substitute 3 times the value of the content of C000 in index register I
② Store content of XW005 in D(3000+I)
③ Add 1 to the content of I and store content of XW010 in D(3000+I)
④ Add a further 1 to the content of I and store content of XW012 in D(3000+I)

Incidentally,

Ⓐ

⊣ P ⊢ is positive transition-sensing contact which becomes ON once only when device Ⓐ changes from OFF to ON (until the instruction is executed in the next scan)

⊣[Ⓐ * Ⓑ→Ⓒ+1•Ⓒ]⊢ is multiplication instruction which multiplies Ⓐ by Ⓑ and stores it in double-length register Ⓒ+1•Ⓒ

⊣[+1 Ⓐ]⊢ is increment instruction which adds 1 to the content of Ⓐ and stores it in Ⓐ

⊣[Ⓐ MOV Ⓑ]⊢ is a data transfer instruction which substitutes the content of Ⓐ in Ⓑ

NOTE

(1) Substitutions of values to index registers and index modification may be carried out any number of times during a program. Therefore, normally, the program will be easier to see if a value substitution to an index register is executed immediately before index modification.
(2) Be careful that the registers do not exceed the address range through index modification. When the results of index modification exceed the address range, the instruction is not executed, and special devices (S0051 and S0064) which indicate 'boundary error' become ON.

As explained before, the main purpose of the index modification is indirect designation of register. However, as the special usage of the index modification, the followings are also possible.

- For CALL and JUMP instructions, indirect designation of the destination address is possible.

```
          I
-[ JUMP N.000 ]-   (If I=5, jump to Label 5)
```

  If indexed destination is not registered, the special devices (S0051 and S006C) become ON. If indexed destination exceeds the range, the special devices (S0051 and S0065) become ON. And both cases, the instruction is not executed.

- For SET and RST instructions, indirect designation of device is possible.

```
          I
-[ SET R0100 ]-   (If I=H005F, set R015F to ON)
```

- For constant operand, the constant value can be modified by the index register.

```
     I
-[ 500 MOV D5000 ]-   (If I=10, 510 is stored in D5000)
```

NOTE

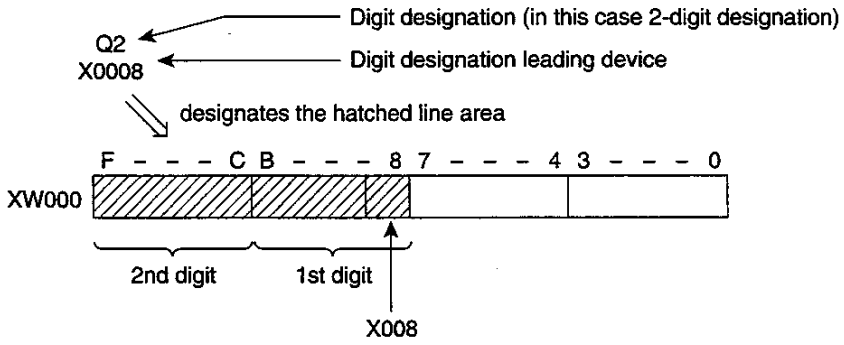
Refer to the Instruction Set manual for the operands to which the index modification is available in each instruction.

3.5 Digit designation

There is a method called 'digit designation' which is a special designation method for register data. 'Digit designation' treats 1 digit (4 bits) of a hexadecimal number as a data unit. It is a method of designation in which a number of digits from the designated devices (bit positions) are made the subject of data operation.

In practice, in the case of the following Example, 2 digits from X0008 (that is to say, the upper 8 bits of XW000) become the subject of data operation.

Example)

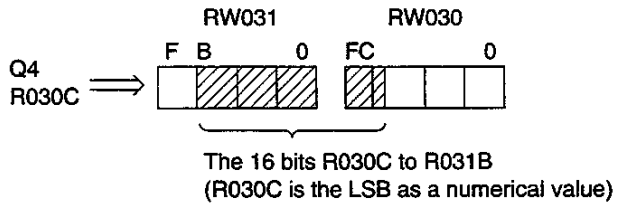

Digit designation (in this case 2-digit designation)
Q2
X0008
Digit designation leading device
designates the hatched line area
F - - - C B - - - 8 7 - - - 4 3 - - - 0
XW000
2nd digit
1st digit
X008

There are 9 types of digit designation - Q0, Q1, ..., Q8 which have the following significations

Q0....makes the designated device 1 bit the subject of data operation
Q1....makes 1 digit (4 bits) started with the designated device the subject of data operation
Q2....makes 2 digits (8 bits) started with the designated device the subject of data operation
Q3....makes 3 digits (12 bits) started with the designated device the subject of data operation
Q4....makes 4 digits (16 bits) started with the designated device the subject of data operation
Q5....makes 5 digits (20 bits) started with the designated device the subject of data operation
Q6....makes 6 digits (24 bits) started with the designated device the subject of data operation
Q7....makes 7 digits (28 bits) started with the designated device the subject of data operation
Q8....makes 8 digits (32 bits) started with the designated device the subject of data operation

In digit designation, when the area designated covers multiple registers, as shown below, the area is designated from the smaller address to the greater address.

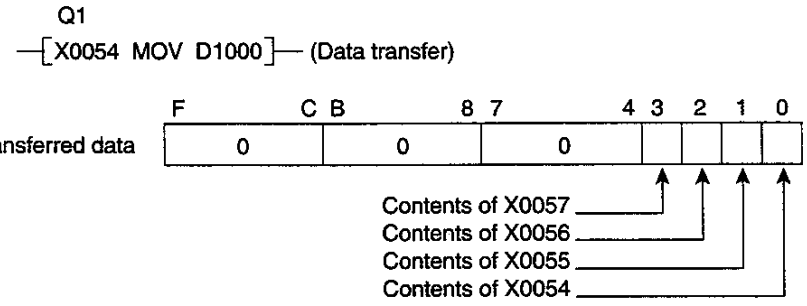
Example)



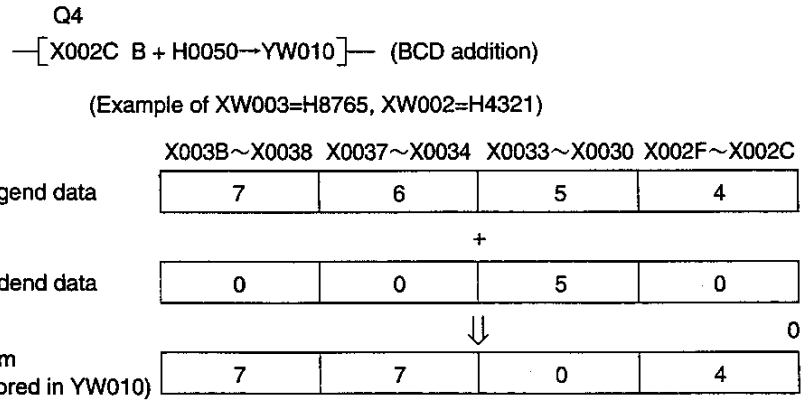
Below, the operation of digit designation is described for the case when digit designation is executed as a source operand (a register for executing an instruction using its data) and the case when digit designation is executed as a destination operand (a register which stores the result of instruction execution).
It is possible to carry out digit designation for both a source operand and a destination operand with 1 instruction.

(1) Digit designation for a source operand
For a single-length (16 bits) operand, Q0 to Q4 are available. The upper digits which are out of the designated digits are regarded as 0.

Example 1)

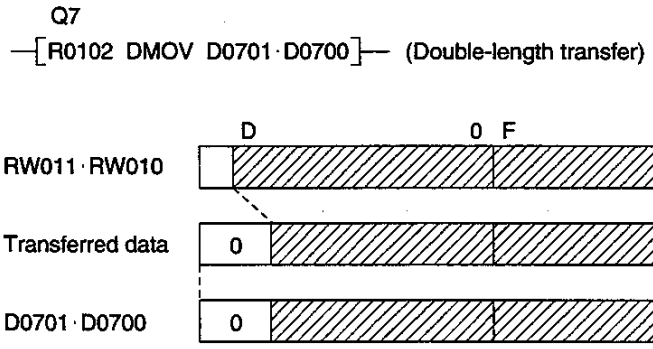


Example 2)



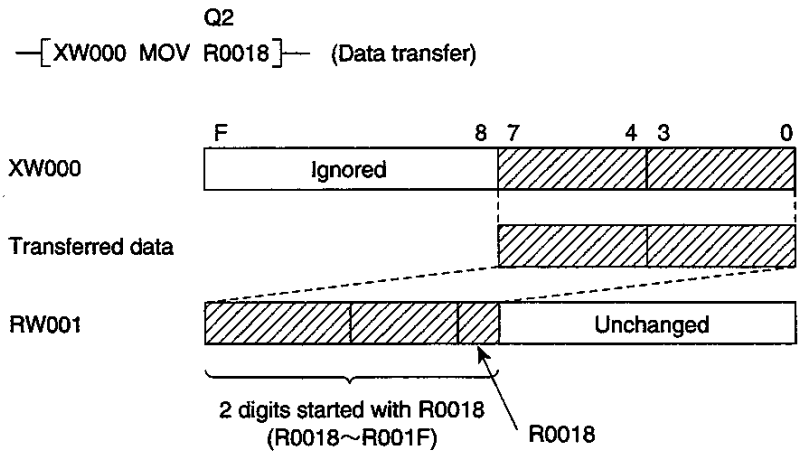
For a double-length (32 bits) operand, all Q0 to Q8 are available.

Example 3)

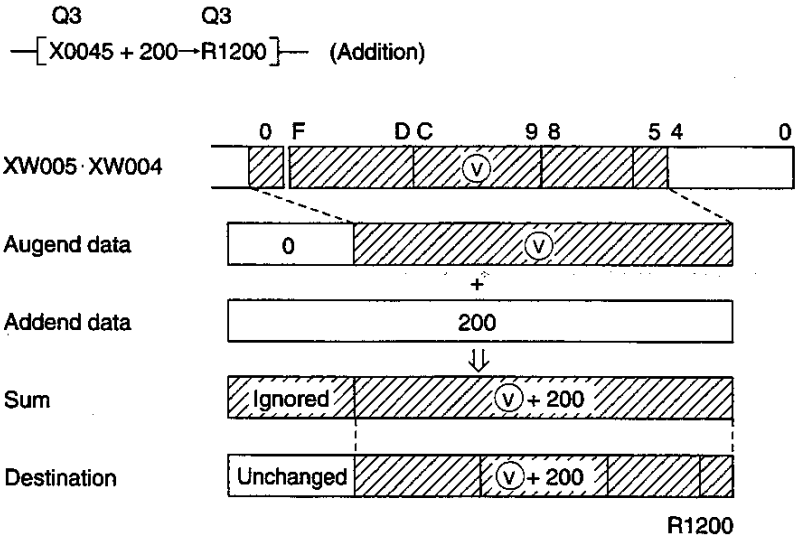


(2) Digit designation for a destination operand
For single-length (16 bits) operand, Q0 to Q4 are available. The result data of the operation is stored in the specified digits of the destination register. The digits which are out of the designated digits are unchanged.

Example 1)



Example 2)



If, XW005=H0077=0000 0000 0111 0111 (binary)
XW004=H182A=0001 1000 0010 1010 (binary)

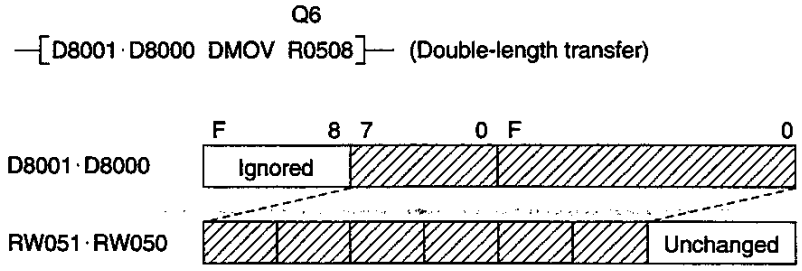
augend data is;
0000 1000 1100 0001 (binary)=H08C1=2241 (decimal)

sum by adding 200;
0000 1001 1000 1001 (binary)=H0989=2441 (decimal)

Therefore, the data below is stored in the 3 digits (12 bits) started with R1200.
1001 1000 1001 (binary)=H989=2441 (decimal)

For a double-length (32 bits) operand, all Q0 to Q8 are available.

Example 3)



NOTE

(1) Be careful that the result of digit designation does not exceed the address range. When the result of digit designation exceeds the address range, the excess portion will be ignored.
(2) A combination of digit designation and index modification can also be used.

Example)

Q1 I → If I=H001C, it signifies the same → Q1 R001C
R0000

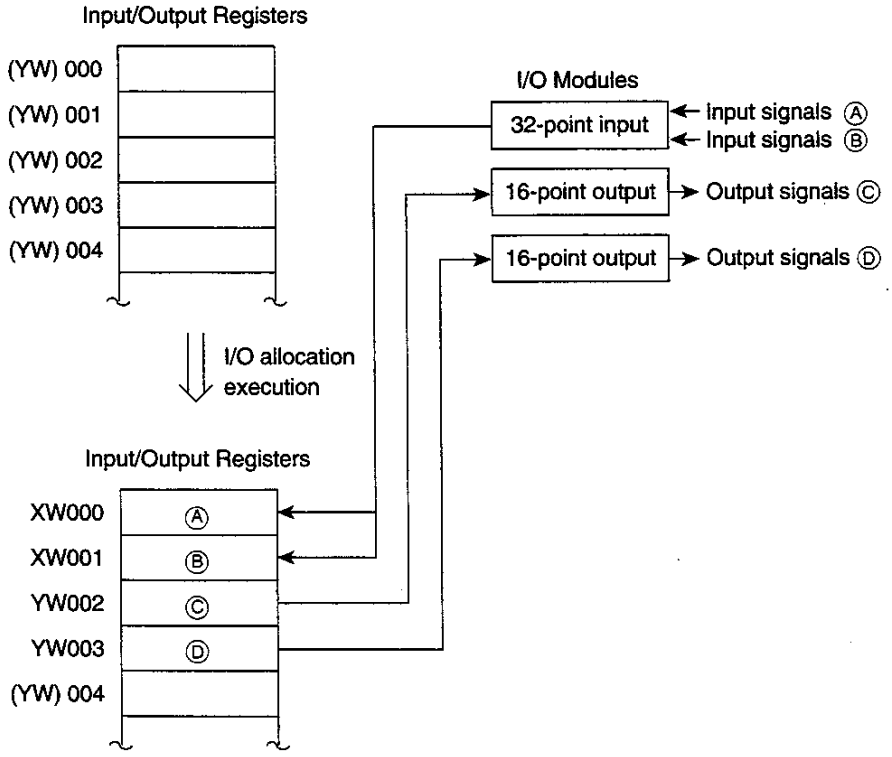
4.1 Overview

The state of external input signals inputted to input modules is read via the input registers/devices (XW/X or IW/I) when scan control is executed. On the other hand, the output data determined in user program execution are outputted to output modules via output registers/devices (YW/Y or OW/O) and outputs from the output modules to external loads are based on these data.

I/O allocation is the execution of mapping between input registers/devices and input modules and of mapping between output registers/devices and output modules. In other words, physical devices called I/O modules are allocated to logic devices called registers/devices.

Input registers/devices and output registers/devices do not use their own independent memory areas. They use a series of memory areas which can be said to be input/output registers/devices (a register address range of 256 words from 000 to 255).

By executing I/O allocation, function type determination is carried out by making addresses allocated to input modules input registers/devices and addresses allocated to output modules output registers/devices.



Note) Addresses not allocated to I/O modules are output (YW) internally.

4.2 Methods of I/O allocation

The execution of I/O allocation can be said in other words to be the carrying out of the registration of I/O allocation information in system information. The T3 CPU checks whether the I/O modules are correctly mounted based on this I/O allocation information when RUN starts-up. Also, at the same time, the correspondence between the input/output registers (XW/YW) and the I/O modules is determined based on this I/O allocation information. On the other hand, the programmer reads this I/O allocation information when communicating with the T3 and recognizes the assignment whether input (XW) or output (YW) for every input/output register address.

There are 2 methods for the registration of I/O allocation information in system information. These are automatic I/O allocation and manual I/O allocation.

The registration of I/O allocation information is only available when the T3 is in the HALT mode but not in the 'memory protect' state (P-RUN).

Automatic I/O allocation

This is a method of causing the T3 to execute the registration of I/O allocation information. It is carried out by selecting and executing the AutoSet command on the I/O allocation screen of the programmer (T-PDS).

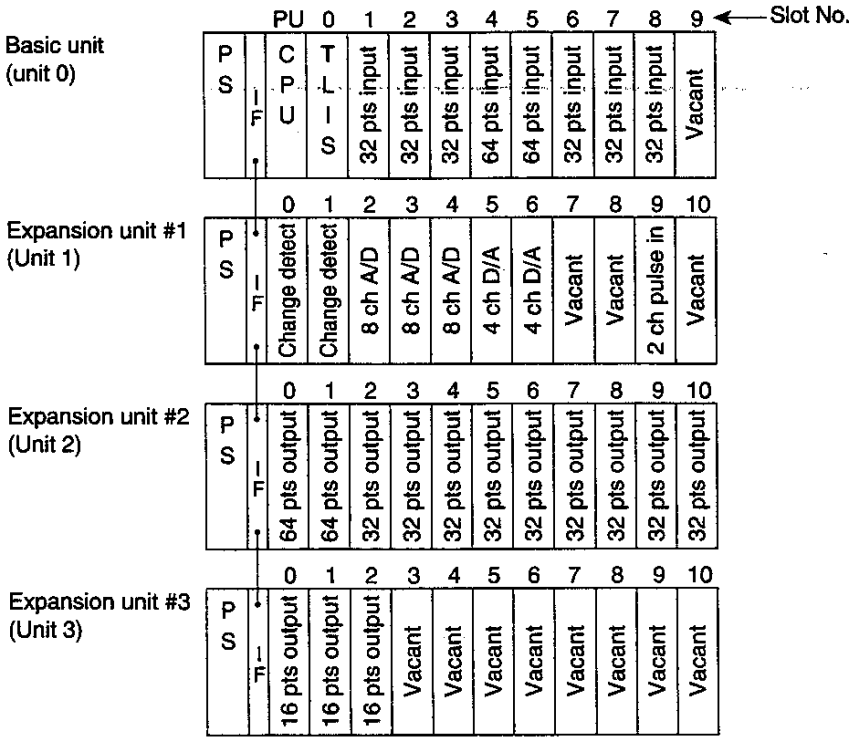
When the automatic I/O allocation is executed, the T3 CPU reads out state of the I/O modules which are mounted (what type of module is mounted in which position) and registers the I/O allocation information.

Each I/O module has one of the module types shown below.

Part No.	Description	Module Type
DI334/334H	32 pts DC input	X 2W
DI335/335H	64 pts DC input	X 4W
IN354/364	32 pts AC input	X 2W
DO333	16 pts DC output	Y 1W
DO334	32 pts DC output	Y 2W
DO335	64 pts DC output	Y 4W
AC363	16 pts AC output	Y 1W
AC364	32 pts AC output	Y 2W
RO364	32 pts relay output	Y 2W
RO363S	16 pts isolated relay output	Y 1W
AD368	8 ch analog input	X 8W
DA364/374	4 ch analog output	Y 4W
CD332	Change detect 8 pts DC input	iX 1W
PI312	2 ch pulse input	iX+Y 2W
AS311	ASCII module	iX+Y 4W
SN321/322/323	TOSLINE-S20	TL-S
MS311	TOSLINE-F10	TL-F

For instance, when automatic I/O allocation is executed with the I/O module mounting state shown below, the CPU reads the I/O module types which are mounted and creates I/O allocation information and it registers it in system information.

- Module Mounting State


PU 0 1 2 3 4 5 6 7 8 9 ← Slot No.
Basic unit (unit 0)
PS I/F CPU TL-IS 32 pts input 32 pts input 32 pts input 64 pts input 64 pts input 32 pts input 32 pts input 32 pts input Vacant
0 1 2 3 4 5 6 7 8 9 10
Expansion unit #1 (Unit 1)
PS I/F Change detect Change detect 8 ch A/D 8 ch A/D 8 ch A/D 4 ch D/A 4 ch D/A Vacant Vacant 2 ch pulse in Vacant
0 1 2 3 4 5 6 7 8 9 10
Expansion unit #2 (Unit 2)
PS I/F 64 pts output 64 pts output 32 pts output 32 pts output 32 pts output 32 pts output 32 pts output 32 pts output 32 pts output 32 pts output 32 pts output
0 1 2 3 4 5 6 7 8 9 10
Expansion unit #3 (Unit 3)
PS I/F 16 pts output 16 pts output 16 pts output Vacant Vacant Vacant Vacant Vacant Vacant Vacant Vacant

- I/O allocation information

Unit 0		Unit 1		Unit 2		Unit 3	
Slot	Module type	Slot	Module type	Slot	Module type	Slot	Module type
PU		0	iX 1W	0	Y 4W	0	Y 1W
0	TL-S	1	iX 1W	1	Y 4W	1	Y 1W
1	X 2W	2	X 8W	2	Y 2W	2	Y 1W
2	X 2W	3	X 8W	3	Y 2W	3	
3	X 2W	4	X 8W	4	Y 2W	4	
4	X 4W	5	Y 4W	5	Y 2W	5	
5	X 4W	6	Y 4W	6	Y 2W	6	
6	X 2W	7		7	Y 2W	7	
7	X 2W	8		8	Y 2W	8	
8	X 2W	9	iX+Y 2W	9	Y 2W	9	
9		10		10	Y 2W	10	

Manual I/O allocation

This is the method by which the user edits the I/O allocation information on the I/O allocation information screen of the programmer (T-PDS) and writes it to the T3. The manual I/O allocation is used in the following cases.

- When carrying out programming in a state in which the I/O modules are not fully mounted
- When it is desired to remove some modules from the subjects of batch input/output processing
- When using the unit base address setting function
- When allocating a specified number of registers to slot left vacant for future addition
- When carrying out offline programming

For manual I/O allocation, module types are set for each slot. The module types which can be set at this time are as shown below. Module types are expressed by combinations of function classifications and numbers of registers occupied. (except for MMR, TL-S and TL-F)

Function classification	Number of registers occupied	Remarks
X	01, 02, 04, 08, 16, 32	Input (batch input/output)
Y	01, 02, 04, 08, 16, 32	Output (batch input/output)
X+Y	02, 04, 08, 16, 32, 64, 128	Input+output (batch input/output)
iX	01, 02, 04, 08, 16, 32	Input (out of batch input/output)
iY	01, 02, 04, 08, 16, 32	Output (out of batch input/output)
iX+Y	02, 04, 08, 16, 32, 64, 128	Input+output (out of batch input/output)
Z	08, 16, 32	
SP	01, 02, 04, 08, 16, 32, 64, 128	Space
MMR	–	Memory type
TL-S	–	For TOSLINE-S20
TL-F	–	For TOSLINE-F10

(1) Allocations to input/output modules are: -X and iX to input modules, Y and iY to output modules and X+Y and iX+Y to input/output mixed modules. The input/output registers which correspond to modules with the designation i attached are not included in batch input/output subjects.
(2) SP is used when allocating an arbitrary number of registers to a vacant slot.
(3) MMR is set in the CPU slot when an IC memory card is used as an expansion memory.
(4) TL-S is allocated to data transmission module TOSLINE-S20.
(5) TL-F is allocated to data transmission module TOSLINE-F10.
(6) Z is not used in the T3.

NOTE

The I/O allocation information can be freely edited and registered by carrying out manual I/O allocation. However, it is necessary that the registered input/output allocation information and the I/O module mounting state should agree for starting-up RUN.
When executing the 'forced RUN' command, operation (RUN-F mode) is possible even if the modules registered in the allocation information are not mounted. However, in this case also, operation cannot be executed when a module of a different type to the registered module is mounted (I/O mismatch).

Unit base address setting function

In manual I/O allocation, the starting register address (input/output registers) of each unit can be set.

The register addresses can be arranged for each unit by using this function. Also, when an I/O module is added in a vacant slot in the future, it is possible to avoid affecting the register addresses of other units.

(Unit base address setting screen on T-PDS)

Unit #0	Unit #1	Unit #2	Unit #3
Top Register No.	Top Register No.	Top Register No.	Top Register No.
[0]	[15]	[35]	[50]

In the case of this screen example, address allocations can be carried out
from XW/YW000 for the basic unit
from XW/YW015 for expansion unit #1
from XW/YW035 for expansion unit #2
from XW/YW050 for expansion unit #3

NOTE

Settings by which latter stage units become lower register addresses cannot be made.

4.3 Register and module correspondence

When I/O allocation information is registered by carrying out automatic I/O allocation or manual I/O allocation, correspondence between registers and modules is automatically determined by the following rules.

(1) In any unit, allocation is the lower address registers are allocated in sequence from the module at the left end.

(2) In a case when the unit base address is not set (it is not set by automatic I/O allocation), the registers are allocated in continuation from the previous stage unit.

(3) A slot for which a module type is not set (any vacant slot in automatic I/O allocation is the same) does not occupy any registers.

(4) The cases of the half size racks (BU315, BU356) also are handled in the same way as standard size rack for I/O allocation, and they are regarded as having slots without settings in the latter portions of the unit. Therefore these portions do not occupy registers.

(5) Slots for which SP (space) is set, output registers are allocated internally by a number of set words.

(6) Modules for which Z, MMR, TL-S and TL-F are set do not occupy input/output registers (XW/YW).

(7) Input/output registers which are not allocated to I/O modules become output registers (YW) in the programming. Thus, they can be used in the same way as auxiliary registers/relays (RW/R).

The following examples show the register allocation when the I/O allocation information is registered.

Example1)

- I/O allocation information

Unit 0		Unit 1		Unit 2		Unit 3	
Base address [ ]		Base address [ ]		Base address [ ]		Base address [ ]	
Slot	Module type	Slot	Module type	Slot	Module type	Slot	Module type
PU		0	iX 1W	0	Y 4W	0	Y 1W
0	X 2W	1	iX 1W	1	Y 4W	1	Y 1W
1	X 2W	2	X 8W	2	Y 2W	2	Y 1W
2	X 2W	3	X 8W	3	Y 2W	3	
3	X 2W	4	X 8W	4	Y 2W	4	
4	X 4W	5	Y 4W	5	Y 2W	5	
5	X 4W	6	Y 4W	6	Y 2W	6	
6	X 2W	7		7	Y 2W	7	
7	X 2W	8		8	Y 2W	8	
8	X 2W	9	iX+Y 2W	9	Y 2W	9	
9		10		10	Y 2W	10	

- Register allocation

Unit 0		Unit 1		Unit 2		Unit 3	
Slot	Register	Slot	Register	Slot	Register	Slot	Register
PU		0	XW022	0	YW058~YW061	0	YW084
0	XW000, XW001	1	XW023	1	YW062~YW065	1	YW085
1	XW002, XW003	2	XW024~XW031	2	YW066, YW067	2	YW086
2	XW004, XW005	3	XW032~XW039	3	YW068, YW069	3	
3	XW006, XW007	4	XW040~XW047	4	YW070, YW071	4	
4	XW008~XW011	5	YW048~YW051	5	YW072, YW073	5	
5	XW012~XW015	6	YW052~YW055	6	YW074, YW075	6	
6	XW016, XW017	7		7	YW076, YW077	7	
7	XW018, XW019	8		8	YW078, YW079	8	
8	XW020, XW021	9	XW056, YW057	9	YW080, YW081	9	
9		10		10	YW082, YW083	10	

Example 2)

- I/O allocation information

Unit 0		Unit 1		Unit 2		Unit 3	
Base address [ ]		Base address [ ]		Base address [100]		Base address [150]	
Slot	Module type	Slot	Module type	Slot	Module type	Slot	Module type
PU		0	X 8W	0	X 4W	0	Y 2W
0	iX 2W	1	X 8W	1	X 4W	1	Y 2W
1	iX 2W	2	X 8W	2	Y 4W	2	Y 2W
2	iX 2W	3	SP 8W	3	Y 4W	3	Y 2W
3	SP 4W	4	Y 4W	4	SP 8W	4	Y 1W
4		5	Y 4W	5	X 2W	5	Y 1W
5	iY 2W	6	Y 4W	6	X 2W	6	Y 1W
6	iY 2W	7	Y 4W	7	X 2W	7	Y 1W
7	iY 2W	8	Y 4W	8	X 2W	8	
8	SP 4W	9		9	X 2W	9	
9		10		10		10	

- Register allocation

Unit 0		Unit 1		Unit 2		Unit 3	
Slot	Register	Slot	Register	Slot	Register	Slot	Register
PU		0	XW020~XW027	0	XW100~XW103	0	YW150, YW151
0	XW000, XW001	1	XW028~XW035	1	XW104~XW107	1	YW152, YW153
1	XW002, XW003	2	XW036~XW043	2	YW108~YW111	2	YW154, YW155
2	XW004, XW005	3	YW044~YW051	3	YW112~YW115	3	YW156, YW157
3	YW006~YW009	4	YW052~YW055	4	YW116~YW123	4	YW158
4		5	YW056~YW059	5	XW124, XW125	5	YW159
5	YW010, YW011	6	YW060~YW063	6	XW126, XW127	6	YW160
6	YW012, YW013	7	YW064~YW067	7	XW128, XW129	7	YW161
7	YW014, YW015	8	YW068~YW071	8	XW130, XW131	8	
8	YW016~YW019	9		9	XW132, XW133	9	
9		10		10		10	

4.4 Network assignment

For the data transmission module (TOSLINE-S20, TOSLINE-F10), the network assignment is necessary in addition to the I/O allocation mentioned before.
The network assignment is the declaration of assignment between the link registers and the scan data memory in the data transmission module.

TOSLINE-S20

The TOSLINE-S20 has 1024 words of scan data memory in the module.
By using the network assignment, T3's link registers (W) are assigned to the scan data memory in units of blocks.
(64 words/block)

Here, the block is not related to the data send block in the TOSLINE-S20. The data transfer direction between the link registers and the scan data memory is determined by T3 CPU for each address, according to the data send block setting in the TOSLINE-S20.

The following 3 types of assignment setting are available.

Setting	Function
Blank	The block of link registers (W) are not assigned to TOSLINE-S20.
LINK	The block of link registers (W) are assigned to TOSLINE-S20. (T3 accesses TOSLINE-S20 for the block)
GLOBAL	Used when 2 TOSLINE-S20s are mounted on the T3, and when the T3 functions as bridge station for the 2 TOSLINE-S20 networks.

Note) Up to 2 TOSLINE-S20s can be mounted on a T3.
In this case, the TOSLINE-S20 nearer to the T3 CPU is regarded as CH1, and the other is CH2.

(1) Example when 1 TOSLINE-S20 is mounted (CH1 only)

- Network assignment example

Block	Corresponding link registers	CH1	CH2
1	W0000~W0063	LINK	
2	W0064~W0127	LINK	
3	W0128~W0191	LINK	
4	W0192~W0255		
5	W0256~W0319		
6	W0320~W0383		
7	W0384~W0447		
8	W0448~W0511		
9	W0512~W0575	LINK	
10	W0576~W0639	LINK	
11	W0640~W0703		
12	W0704~W0767		
13	W0768~W0831		
14	W0832~W0895		
15	W0896~W0959		
16	W0960~W1023		

- Data transfer direction

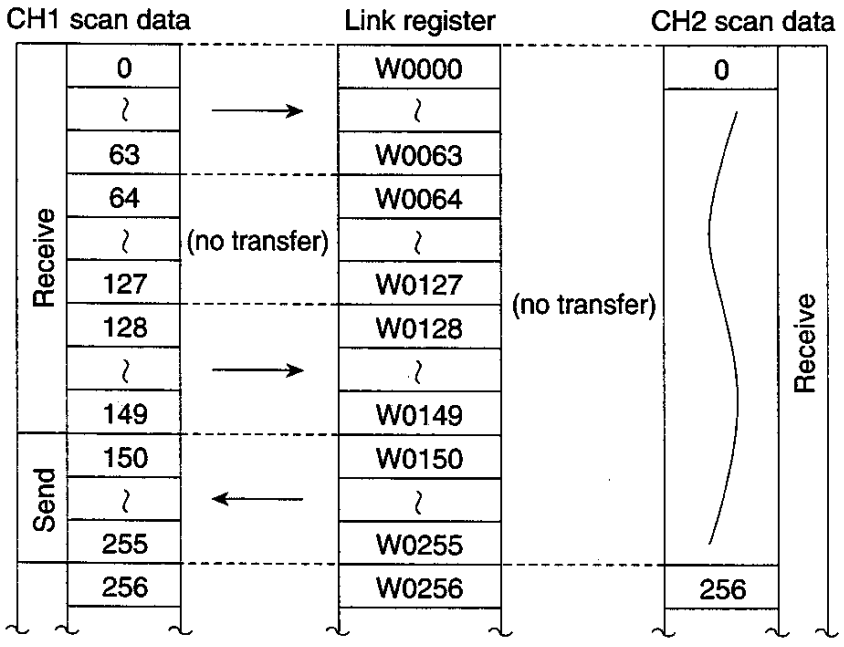
Link register	Data transfer direction	CH1 scan data	
W0000	→	0	Send
≀	→	≀	Send
W0149	→	149	Send
W0150	←	150	Receive
≀	←	≀	Receive
W0191	←	191	Receive
W0192	(no transfer)	192	Receive
≀	(no transfer)	≀	Receive
W0511	(no transfer)	511	Receive
W0512	←	512	Receive
≀	←	≀	Receive
W0639	←	639	Receive
W0640	(no transfer)	640	Receive
≀	(no transfer)	≀	Receive
W1023	(no transfer)	1023	Receive

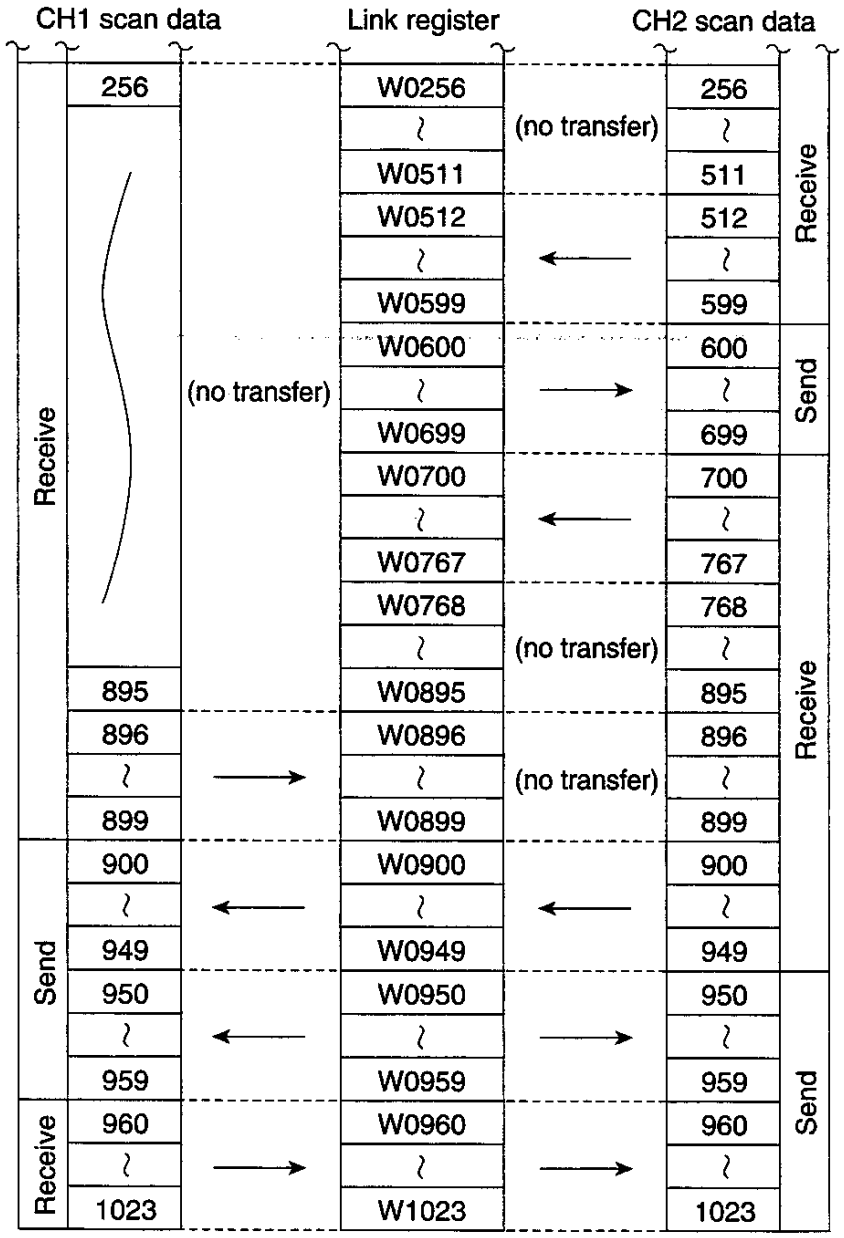
(2) Example when 2 TOSLINE-S20s are mounted (CH1, CH2)

- Network assignment example

Block	Corresponding link registers	CH1	CH2
1	W0000~W0063	LINK	
2	W0064~W0127		
3	W0128~W0191	LINK	
4	W0192~W0255	LINK	
5	W0256~W0319		
6	W0320~W0383		
7	W0384~W0447		
8	W0448~W0511		
9	W0512~W0575		LINK
10	W0576~W0639		LINK
11	W0640~W0703		LINK
12	W0704~W0767		LINK
13	W0768~W0831		
14	W0832~W0895		
15	W0896~W0959	GLOBAL	
16	W0960~W1023	GLOBAL	

- Data transfer direction





NOTE

(1) In the GLOBAL setting block, the scan data is read from the receive setting CH, and transferred to the link registers and the send setting CH.
(2) In the GLOBAL setting block, if both CHs are set as send, the link registers data is transferred to both CHs.
(3) In the GLOBAL setting block, if both CHs are set as receive, the GLOBAL setting CH's data is transferred to the link registers.
(4) For one block, settings for CH1 and CH2 should not be duplicated. If duplicated, CH2 setting is ignored.

TOSLINE-F10

The TOSLINE-F10 has 32 words of scan data memory in the module. Up to 8 TOSLINE-F10 can be mounted on a T3. In this case, the TOSLINE-F10 nearer to the T3 CPU is assigned in sequence from CH1 to CH8.

For the TOSLINE-F10, set LINK for all existing CHs by the network assignment. By this setting, the link registers (LW) are assigned to the TOSLINE-F10 in units of 32 words from the lowest address.

- Network assignment when 4 TOSLINE-F10s are mounted

CH	Setting	Assigned link register (LW)
1	LINK	LW000~LW031
2	LINK	LW032~LW063
3	LINK	LW064~LW095
4	LINK	LW096~LW127
5		–
6		
7		
8		

The data transfer direction between the link registers (LW) and the scan data in the TOSLINE-F10 is determined by T3 CPU, according the TOSLINE-F10 network configuration.

NOTE

For details of the data transmission modules (TOSLINE-S20, TOSLINE-F10), see separate manuals for them.

5.1 Overview

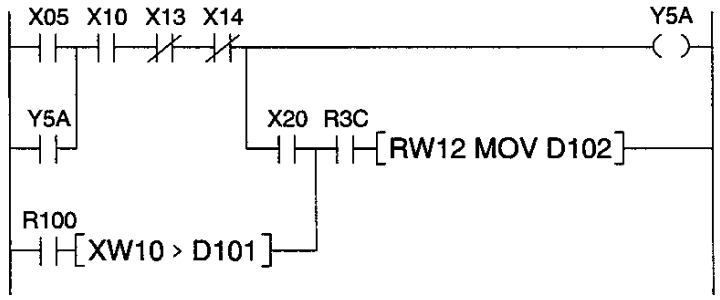
The T3 supports 2 types of programming language for the user programs-ladder diagram and SFC. Multiple programming languages can be used in mixed by a single user program by separating blocks of the program. Thus, the optimum program configuration for the control functions can be achieved.

(1) Ladder Diagram
This is the language which is core programming language for the T3. The program is configured by a combination of relay symbols and function blocks. This language is suitable for logic control.

Relay Symbols .....These are NO contact, NC contact, coil, etc.

Function Blocks.....These are box type instructions which express single functions. They can be freely positioned in a ladder diagram network by treating them in a similar way to relay contacts. The output of one function block can be connected to the input of another function block.

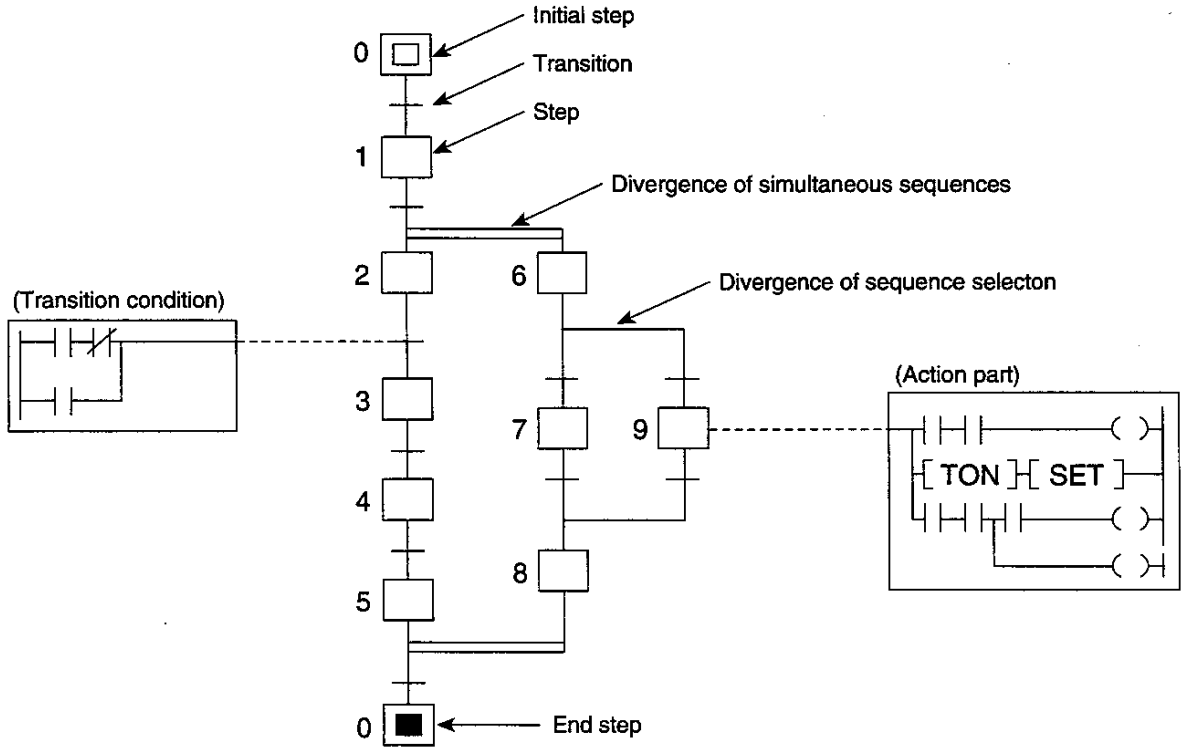
Example)



(2) SFC (Sequential Function Chart)

This is a programming language suitable for process stepping control (sequential control). Also, it is a language which makes the flow of control easy to see. Therefore, it is effective for program maintenance and standardization. SFC program is composed of structure part which shows the flow of control, action parts which show the operation of each step and transition condition parts which enable the process to advance. Action parts and transition condition parts are produced by ladder diagram. SFC can be considered as an execution control element for making a program easier to see by arranging the control processes and conditions rather than a single programming language.

(SFC Structure)



The flow of control advances downward from the initial step and, when it reaches the end step, it returns to the initial step. A step corresponds to an operational process, and there is an action part corresponding to each step. The condition of shifting from one step to the next is called 'transition', and there is a transition condition corresponding to each transition. When the immediately preceding step of a transition is in the active state and the transition condition is ON, the state of the immediately preceding step is changed to inactive and the next step becomes active.

The following Table shows the programming languages which are usable for each program type/part.

Program type/part	Ladder diagram	SFC
Main program	○	○
Sub-program	○	○
Interupt program	○	×
Sub-routine	○	×*
SFC action program part	○	×*
SFC transition condition part	○	×

○: Usable
×: Not usable

*) SFC can be made an hierarchical structure (other SFC can be made to correspond to 1 step of SFC). In this case a macro-step (equivalent to an SFC sub-routine) is used.

5.2 Ladder diagram

Mixed use can be made of the two types of programming language, ladder diagram and SFC in the T3. However, of these, ladder diagram is the basic language which must be present in the user program.

Here, the structure, execution sequence and general items of ladder diagram instructions are explained for ladder diagram programs.

As explained before, a user program is registered by every functional type which is called a program type. Furthermore, in each program type the user program is registered by one or a multiple of units called 'blocks'.

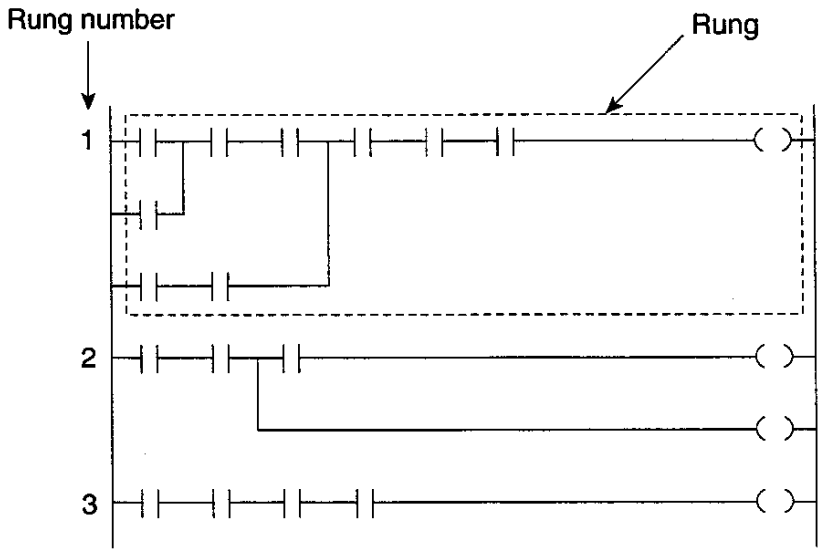
Program Types..... { Main program, sub-program #1 - #4, timer interrupt program, I/O interrupt programs #1 - #8, sub-routine

Blocks .....Blocks 1-256 (1 language/1 block).

When commencing programming in a block to be newly registered, that program is designated by the language which is used (this is called 'language designation').

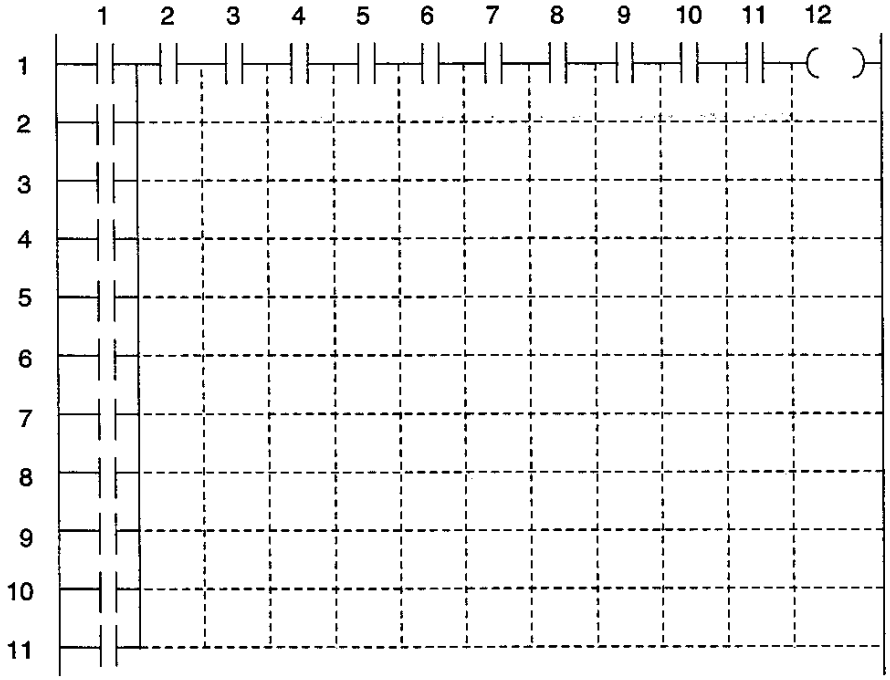
However, in the case of ladder diagram, the operation of language designation is not required (the default is ladder diagram).

The ladder diagram program in any one block is registered/arranged by units called 'rung'. A rung is defined as 1 network which is connected to each other, as shown below.



The rung numbers are a series of numbers (decimal numbers) starting from 1, and rung numbers cannot be skipped. There is no limit to the number of rungs.

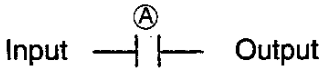
The size of any one rung is limited to 11 lines x 12 columns, as shown below.



Ladder diagram is a language which composes programs using relay symbols as a base in an image similar to a hard-wired relay sequence. In the T3, in order to achieve an efficient data-processing program, ladder diagram which are combinations of relay symbols and function blocks are used.

Relay Symbols.....These are NO contact, NC contact, coil and contacts and coils to which special functions are given. Each of these is called an 'instruction'. (Basic ladder instructions)

Example) NO contact



When device Ⓐ is ON, the input side and the output side become conductive.

Viewed from the aspect of program execution, the operation is such that when the input is ON and the content of device Ⓐ is also ON, the output will become ON.

Function Blocks.....These are expressed as boxes which each show 1 function. As types of function, there are data transfers, the four arithmetic operations, logic operations, comparisons, and various mathematical functions. Each of these is called an 'instruction'. (Function instructions)

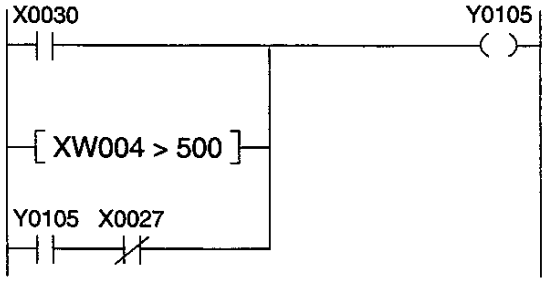
In a function block there are 1 or more inputs and 1 output. When a certain condition is satisfied by the input state, a specified function is executed and the ON/OFF of the output is determined by the result of execution.

Example 1) Addition

Input -[Ⓐ + Ⓑ → Ⓒ]- Output

When the input is ON the content of register Ⓐ and the content of register Ⓑ are added and the result is stored in register Ⓒ. The output becomes ON if an overflow or an underflow is generated as the result of the addition.

Example 2) Combination of Relay Symbols and Function Blocks



When X0030 is ON or the content of XW004 exceeds 500, Y0105 becomes ON. Y0105 stays on even if X0030 is OFF and the content of XW004 is 500 or less, then Y0105 will become OFF when X0027 becomes ON.

NOTE

(1) A function block can be regarded as a contact which has a special function. By carefully arranging the function blocks in the order of execution of instructions, complex control functions can be achieved by an easily understandable program.
(2) A list of ladder diagram instructions is shown in Section 5.5. For the detailed specifications of each instruction, see the separate volume, 'Instruction set Manual'.

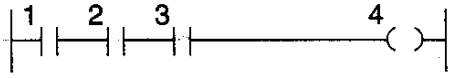
Instruction execution sequence

The instructions execution sequence in a block composed by ladder diagram are shown below.

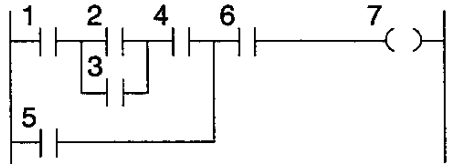
(1) They are executed in the sequence rung1, rung2, rung3... through to the final rung in the block (in the case of a block with an END instruction, through to the rung with the END instruction).

(2) They are executed according to the following rules in any one rung.

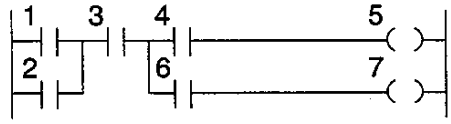
① When there is no vertical connection, they are executed from left to right.



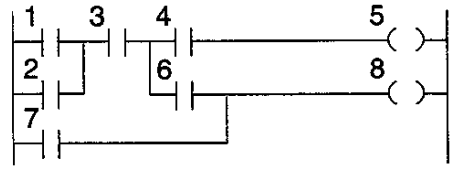
② When there is an OR connection, the OR logic portion is executed first.



③ When there is a branch, they are executed in the order from the upper line to the lower line.



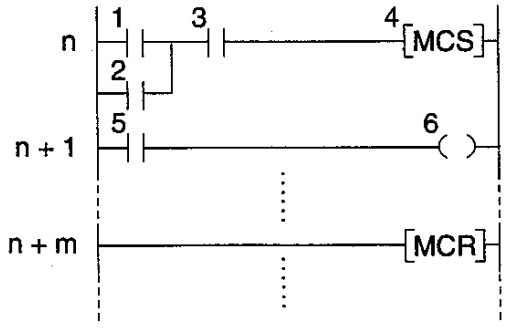
④ A combination of ② and ③ above



The instructions execution sequence in which function instructions are included also follows the above rules. However, for program execution control instructions, this will depend on the specification of each instruction.

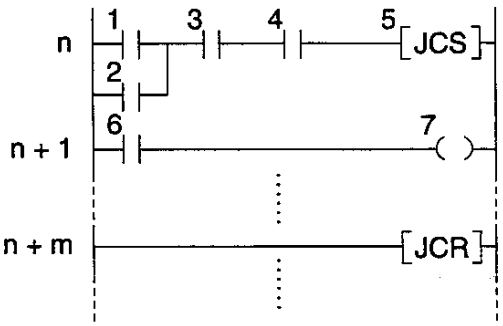
The following show the execution sequences in cases in which program execution control instructions are used.

- Master Control (MCS/MCR, MCSn/MCRn)



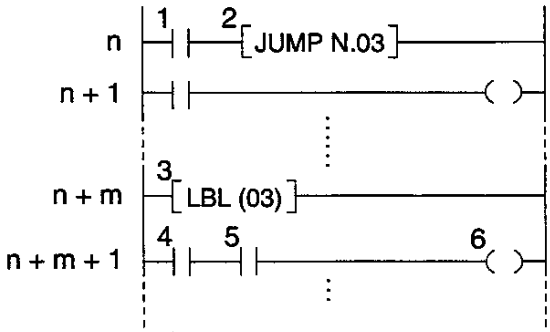
When the MCS input is ON, execution is normal. When the MCS input is OFF, execution is by making the power rail from the rung following MCS to the rung of MCR OFF (the execution sequence is the same).

- Jump Control (JCS/JCR)



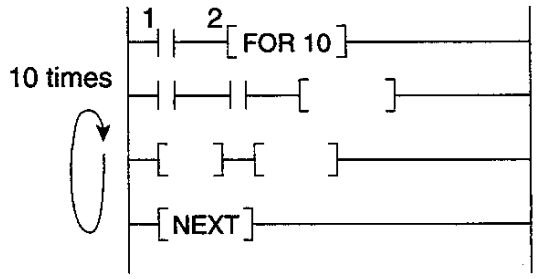
When the JCS input is ON, the instructions from the rung following JCS to the rung of JCR are read and skipped at high speed (instructions are only read and not executed). When the JCS input is OFF, execution is normal.

- Conditional Jump (JUMP/LBL)



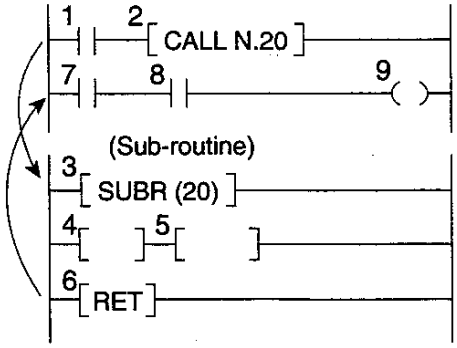
When the JUMP instruction input is ON, execution shifts to the rung following the LBL instruction with the corresponding label number (03 in the example on the left)(the numbers in the diagram on the left are the execution sequence at this time). When the JUMP instruction input is OFF, execution is normal.

- Repeat (FOR/NEXT)



When the FOR instruction input is ON, the instructions between FOR and NEXT are repeatedly executed the designated number of times (10 times in the example on the left), and when the designated number of times is reached, execution is shifted to the rung following the NEXT instruction.
When the FOR instruction input is OFF, execution is normal.

- Sub-Routine (CALL/SUBR/RET)



When the CALL instruction input is ON, execution is shifted to the rung following the SUBR instruction with the corresponding sub-routine number (20 in the example in the left). When the RET instruction is reached, execution is returned to the instruction following the CALL instruction (the numbers in the diagram on the left are the execution sequence at this time). When the CALL instruction input is OFF, execution is normal.

General information on ladder diagram instructions

The general information required for designing programs with ladder diagram are listed below.

(1) In all program types, it is necessary to create at least one block by ladder diagram. In other words, the ends of the main program and each sub-program are judged by ladder diagram END instruction. Also, the end of each interrupt program is judged by a ladder diagram IRET instruction. Furthermore, it is necessary to compose the entry to and exit from a sub-routine by the ladder diagram SUBR instruction and RET instruction.

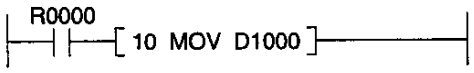
(2) The group of instructions which includes the timer instructions (4 types), counter instruction, jump control instruction, master control instruction and END instruction in the relay symbol type instructions is called the 'basic ladder instructions'.

(3) Instructions other than the basic ladder instructions are called 'function instructions'. The function instructions have respective individual function numbers (FUN No.). Also,even if instructions have the same function number, selection of the execution conditions is possible as shown below. (There are some instructions which cannot be selected)

Normal.....Executed every scan while the instruction input is ON.
Edged.....Executed only in the scan in which the instruction input changes from OFF to ON.

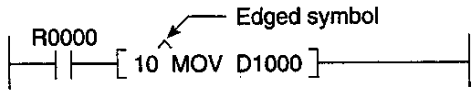
Example) Data Transfer Instruction

Nomal



The MOV instruction (substitute 10 in D1000) is executed every scan while R0000 is ON.

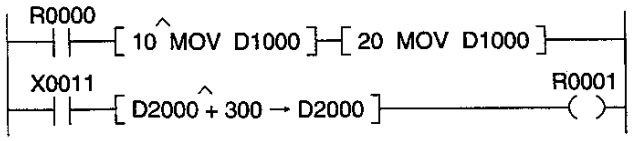
Edged



The MOV instruction (substitute 10 in D1000) is executed only in the scan in which R0000 changes from OFF to ON.

Any instructions cannot be positioned after (to the right of) a edged function instruction.

Example)

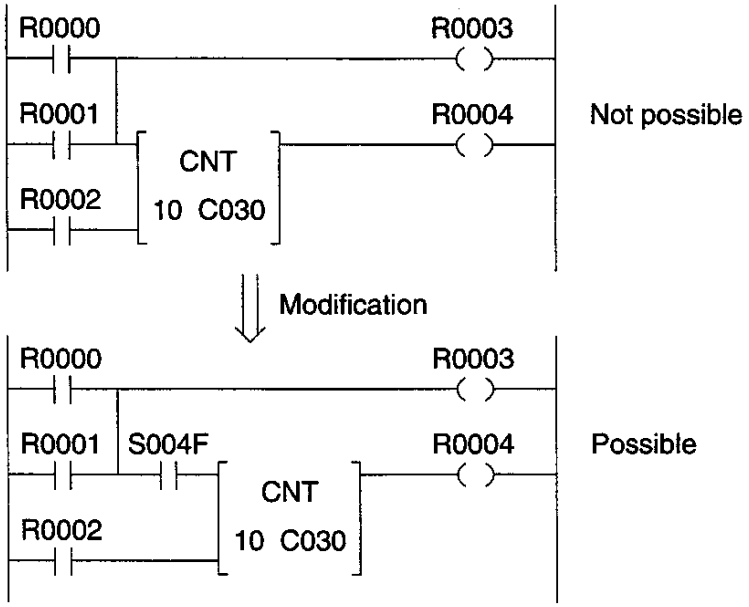


Neither of these two rungs can be created.

(4) The number of steps required for one instruction differs depending on the type of instruction. Also, even with the same instruction, the number of steps occupied varies depending on whether digit designation is used in the operand, a constant or a register is used in a double-length operand, etc. (1-10 steps/1 instruction).
Also, basically step numbers are not required for vertical connection lines and horizontal connection lines.

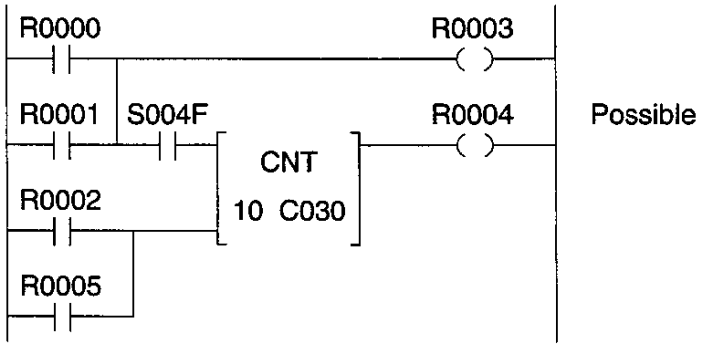
(5) In a instruction which has multiple inputs, a vertical connection line cannot be placed immediately before an input. In this case, insert a dummy contact (such as the NO contact of special relay S004F which is always ON) immediately before the input.

Example)



The above arrangement is not required for the lowest input of multiple inputs.

Example)



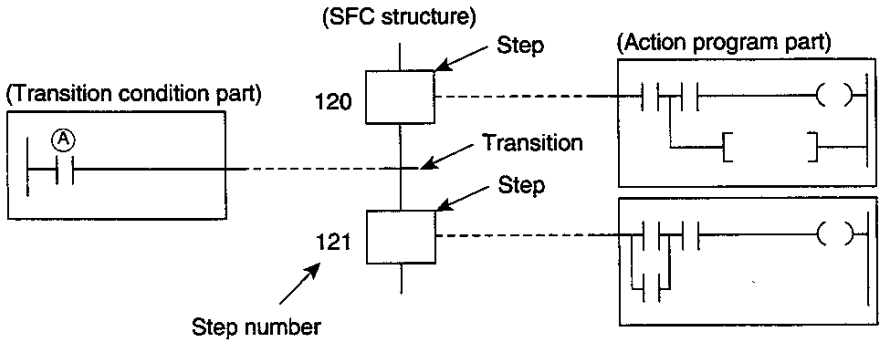
5.3 SFC

SFC is the abbreviation of Sequential Function Chart. This is a programming language suitable for process stepping control (sequential control). In the T3, the following function can be used in the SFC.

- Jump .....Moves the active state to an arbitrary step when a jump condition is satisfied.
- Step with waiting time .....Even if the transition condition is satisfied, step transition is not carried out until a set time has elapsed. (Wait step)
- Step with alarm .....When transition to the following step is not carried out even if the set time has elapsed, the designated alarm device becomes ON. (Alarm step)

SFC can be used in the main program and in the sub-programs. Here the overall composition of SFC, the elements of SFC and notes on program creation are described.

An SFC program is composed of SFC structure,action program parts and transition condition parts.



An SFC structure regulates the flow of the control operation and has steps and transitions as its basic elements. A step is expressed by one box, as shown above. Each step has its own step number. Also, corresponding action program parts are annexed 1 to 1 to steps.

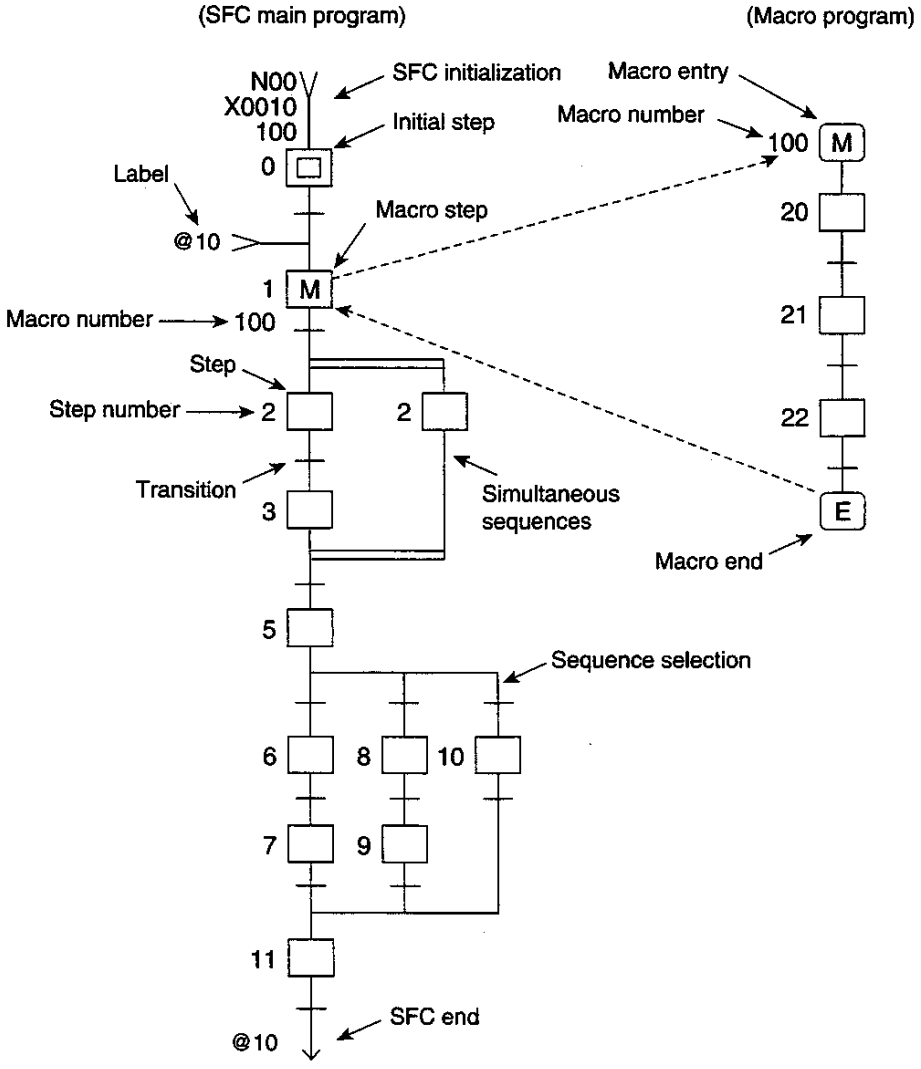
Steps have the two states of active and inactive. When a step is active, the power rail of the corresponding action program will be ON. When a step is inactive, the power rail of the corresponding action program will be OFF.

On the other hand, a transition is located between step and step, and expresses the conditions for transition of the active state from the step immediately before (upper step) to the following step (lower step). Corresponding transition conditions are annexed 1 to 1 to transitions.

For instance, in the diagram above, when step 120 is active, the action program power rail corresponding to step 120 becomes ON. In this state, when device Ⓐ becomes ON, the transition conditions are satisfied, and step 120 becomes inactive and step 121 becomes active. In accompaniment to this, the action program power rail corresponding to step 120 becomes OFF (executed as power rail OFF), and the action program power rail corresponding to step 121 becomes ON.

Overall configuration

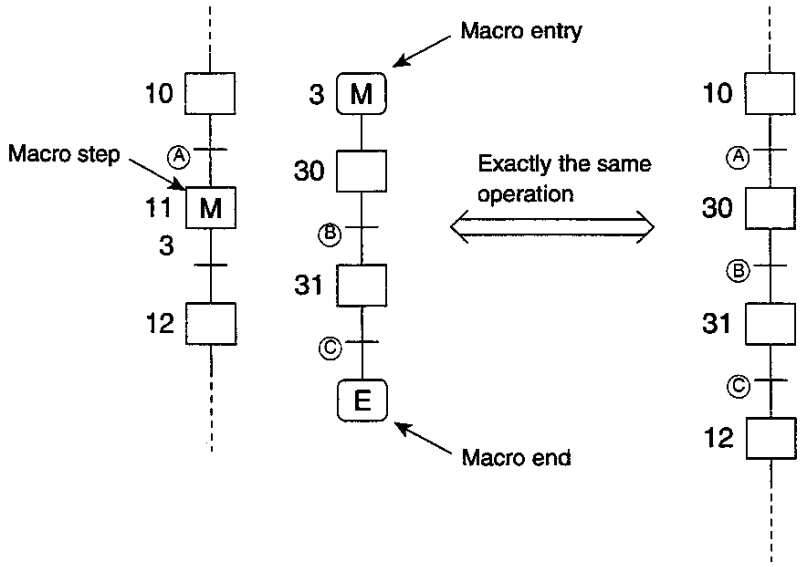
The following illustrates the overall configuration of an SFC Program.



The overall SFC program can be considered as divided into an SFC main program and a macro program.

The SFC main program has an initial step in its structure, and has an SFC end or an End step in its bottom. In the T3, a maximum of 64 SFC main programs can be created.

On the other hand a macro program is a sub-sequence which starts from 'macro entry' and finishes at 'macro end'. Each macro program has its own macro number, and corresponds 1 to 1 to macro steps which are present in the SFC main program or other macro programs. Macro programs are used for rendering the program easy to see by making the SFC program an hierarchical structure. In all, 128 macro programs can be created.



NOTE

(1) Macro steps can be used in macro programs (SFC multi-level hierarchy). There is no limit to the number of levels.
(2) Macro programs and macro steps must correspond 1 to 1.That is to say, macro steps designated with the same macro number cannot be used in multiple locations.
(3) Macro program should be programmed in the following location than the SFC main program/macro program which has the corresponding macro step. (in upper numbered block)

SFC programming becomes possible by designating blocks and then selecting SFC by language designation.

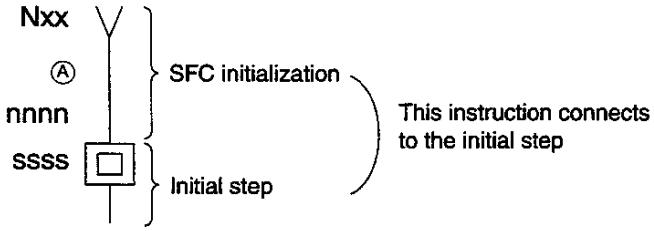
Only one SFC main program or one macro program can be created in 1 block. (1 SFC/block)

Also, the maximum number of SFC steps per block is 128.

SFC elements The following is a description of the elements which compose an SFC program.

(1) SFC Initialization
This is the function which starts-up (makes active) the designated initial step by making the steps in a designated area inactive. Either of the two methods of an SFC instruction or a ladder diagram instruction is used. One SFC initialization is required for 1 SFC main program.

① SFC Instruction



Operands: xx = Program number (0-63)
Ⓐ =Start-up device (except T.and C.)
nnnn = Number of initialized steps (1-4096)

Function: When the device (with the exception of a timer device or a counter device) designated by Ⓐ changes from OFF to ON, the number of steps following the initial step (ssss) which are designated by nnnn (from step number ssss to ssss + nnnn -1), are made inactive, and the initial step (ssss) is made active.

② Ladder Diagram Instruction (FUN 241)

Input ⊣[SFIZ (nnnn) ssss]⊢ Output

Operands: nnnn=Number of initialized steps (1-4096)
ssss = Step number of initial step (0-4095)

Function: When the input changes from OFF to ON, the number of steps designated by nnnn from the step number designated by ssss (from step number ssss to ssss + nnnn -1) are made inactive, and the initial step designated by ssss is made active.

(2) Initial Step

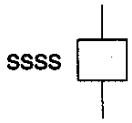
This is the step which indicates the start of an SFC main program. It has its own step number and can have an action program part which corresponds 1 to 1.

Only 1 initial step can be programmed in 1 block.

ssss = Step number (0-4095)

(3) Step

This expresses one unit of contral steps. The step has its own step numbers and has an action program part which corresponds 1 to 1.



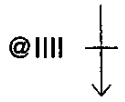
ssss = Step number (0-4095)

(4) Transition

This expresses the conditions for shifting the active state from a step to the following step. Transition has a transition condition part which corresponds 1 to 1.

(5) SFC End

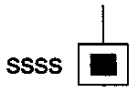
This expresses the end of an SFC main program. An SFC main program requires either this 'SFC end' or the 'end step' of (6). The 'SFC end' has a transition condition which corresponds 1 to 1 and a return destination label number. When transition condition is satisfied with the step immediately before being in the active state, the step following the designation label is made active with making the step immediately before inactive. (This is the same operation as that described in 'SFC jump' below).



IIII = Label number (0-1023)

(6) End Step

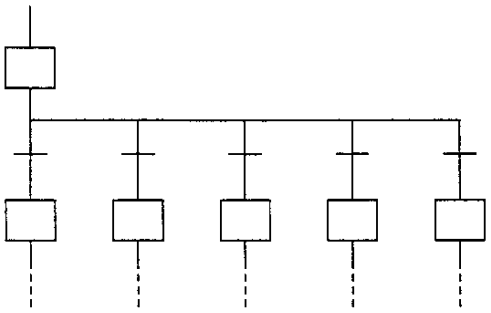
This expresses the end of an SFC main program. An SFC main program requires either this 'end step' or the 'SFC end' of (5). The end step has the same step number as the initial step. When the immediately preceding transition condition is satisfied, the initial step returns to the active state.



ssss = Initial step number (0-4095)

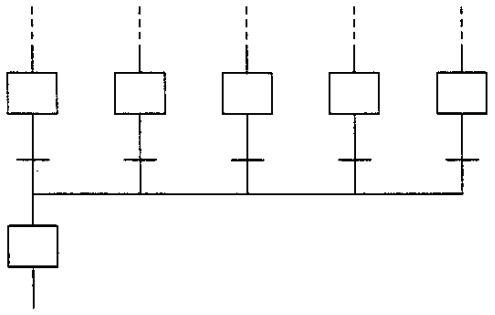
(7) Sequence Selection (divergence)

This transfers the active state to 1 step in which the transition condition is satisfied out of multiple connected steps. When the transition conditions are satisfied simultaneously, the step on the left has priority. (The number of branches is a maximum of 5 columns).

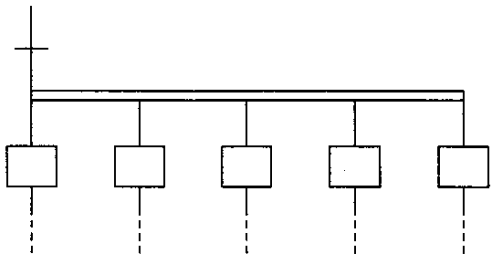


(8) Sequence Selection (convergence)

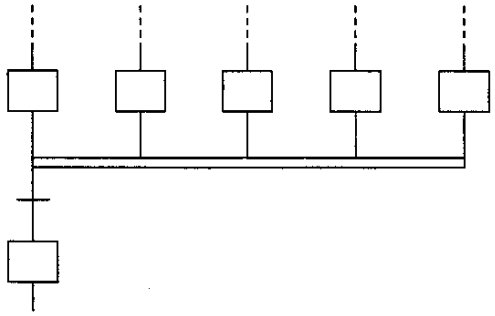
This collects into 1 step the paths diverged by above (7).



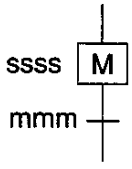
(9) Simultaneous Sequences (divergence)
After the immediately preceding transition condition is satisfied, this makes all the connected steps active. (The number of branches is a maximum of 5 columns).



(10) Simultaneous Sequences (convergence)
When all the immediately preceding steps are active and the transition condition is satisfied, this shifts the active state to the next step.



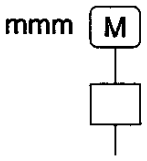
(11) Macro Step
A macro step corresponds to one macro program. When the immediately preceding transition condition is satisfied, this shifts the active state to macro program with the designated macro number. When the transition advances through the macro program and reaches the macro end, the active state is shifted to the step following the macro step. A macro step is accompanied by a dummy transition which has no transition condition (always true).



ssss = Step number (0-4095)
mmmm = Macro number (0-127)

(12) Macro Entry

This expresses the start of a macro program. The macro entry has no action program. Steps are connected below the macro entry. Only 1 macro entry can be programmed in 1 block.



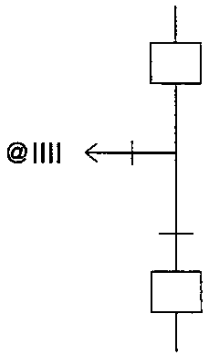
mmm = Macro number (0-127)

(13) Macro End

This expresses the end of a macro program. Macro end has a transition condition which corresponds 1 to 1, and returns to the corresponding macro step when this transition condition is satisfied.

(14) SFC Jump

This expresses a jump to any arbitrary step. Jump has a jump condition which corresponds 1 to 1, and jump destination label numbers. When the transition condition is satisfied, the active state jumps to the step following the designated label. When the jump transition condition and the transition condition for the following step are simultaneously satisfed, jump has priority.

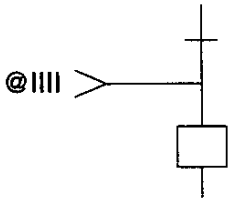


IIII = Label number (0-1023)

'SFC Jump' is located immediately after a step. SFC Jumps with the same label number may be present in multiple locations.

(15) SFC Label

This expresses the return destination from an 'SFC end' and the jump destination from a 'SFC jump'. Label is located immediately after transitions.



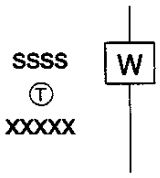
IIII = Label number (0-1023)

NOTE

Note that, when SFC label corresponding to SFC end or SFC jump is not present, or when SFC labels with the same label number are present in multiple locations, an error will occur when RUN starts-up.

(16) Wait Step

This is a step which measures the time after becoming active, and does no execute transition even if the following transition condition is satisfied, until a set time has elapsed. It has an action program corresponding 1 to 1.

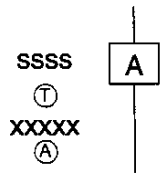


ssss = Step number (0-4095)
Ⓣ= Timer register (T000-T511)
xxxxx = Set time (0-65535)

(Note) T000-T063 are 0.01 second timers
T064-T511 are 0.1 second timers

(17) Alarm Step

This is a step which measures the time after becoming active, and when the transition condition is not satisfied within a set time, switches ON a designated alarm device. It has an action program corresponding 1 to 1. When the transition condition is satisfied and the alarm step becomes inactive, the alarm device also becomes OFF.



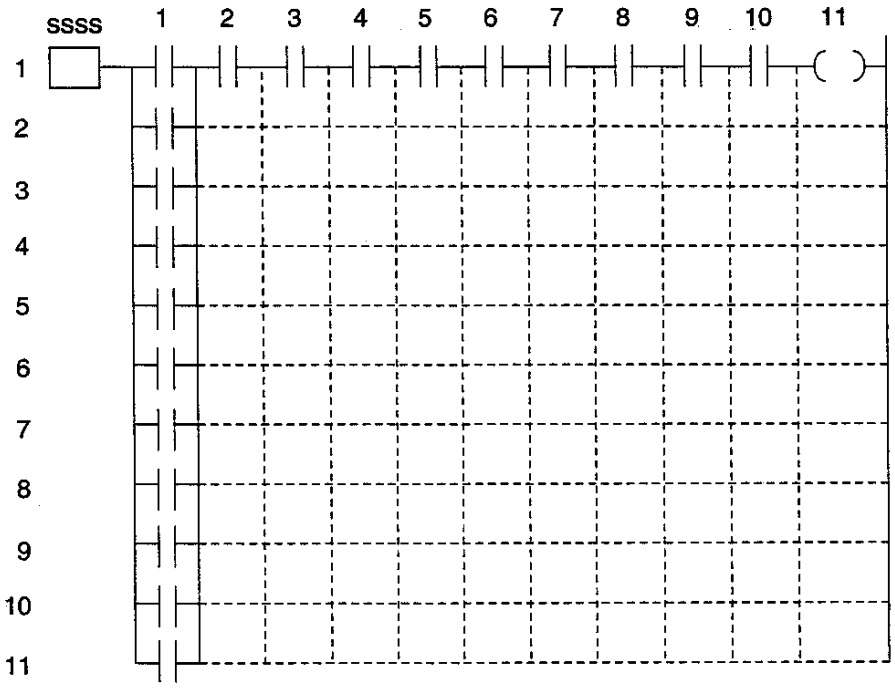
ssss = Step number (0-4095)
Ⓣ= Timer register (T000-T511)
xxxxx = Set time (0-65535)
Ⓐ= Alarm device (other than X, T., C.)

(Note) T000-T063 are 0.01 second timers
T064-T511 are 0.1 second timers

Action program and transition condition

The action program corresponds to 1 step, and the transition condition corresponds to 1 transition.
These are programmed by ladder diagram.

(1) Action Program
The size of 1 action program is 11 lines × 11 columns as shown below, and the number of instruction steps is a maximum of 121 steps.

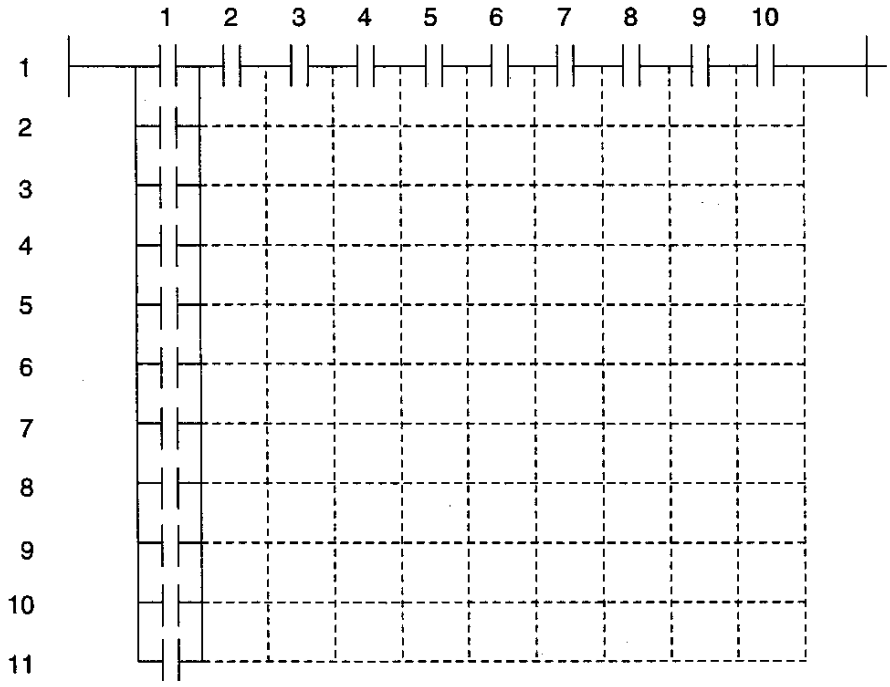


In a case when a larger size than the above is required as an action program, a sub-routine is used. (CALL instruction)
Even if there is no action corresponding to a step, this does not affect SFC operation. In this case, the step becomes a dummy step (a step which waits only the next transition condition will be satisfied).

In programming, by designating the step on the SFC screen and selecting the detail display mode, the monitor/edit screen for the action program corresponding to that step will appear.
In the case when the content of the action program is only 1 instruction out of SET, RST, coil, invert coil, positive pulse coil and negative transition-sensing coil, direct editing can be carried out without putting up the detail display screen. See the programmer (T-PDS) operation manual in a separate volume for this operation.

(2) Transition Condition

The size of 1 transition condition is 11 lines × 10 columns, and the number of instruction steps is a maximum of 110 steps.



When there is no transition condition corresponding to a certain transition, that transition condition is always regarded as true. (Dummy transition)

In programming, by designating the transition on the SFC screen and selecting the detail display mode, the monitor/edit screen for the transition condition corresponding to that transition will appear. In the case when the content of the transition condition is only 1 instruction of NO contact or NC contact, direct editing can be carried out without putting up the detail display screen. See the programmer (T-PDS) operation manual in a separate volume for this operation.

NOTE

The following execution control instructions cannot be used in action programs and transition conditions.

- Jump (JSC/JCR, JUMP/LBL)
- Master control (MCS/MCR, MCSn/MCRn)
- End (END)
- FOR-NEXT (FOR/NEXT)

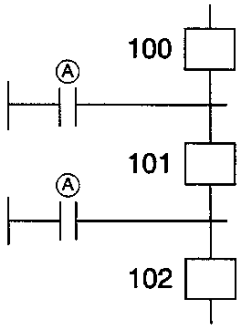
Also, the invert contact and various coil instructions cannot be used in transition conditions.

Execution system The following shows the concept of the execution system in one SFC program.

(1) In one scan, evaluation of the transition condition, the step transition processing and the execution of the action program are sequentially operated.

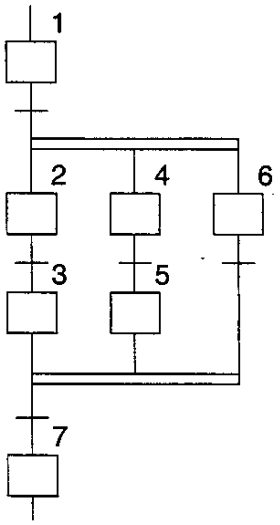
(2) Evaluation of the transition condition means the execution of the transition condition connected to an active step and carrying out a check for transition condition establishment. At this time, since evaluation is made only for active step, there are no multiple step transitions by 1 scan in consecutively connected steps.

For instance, as shown in the diagram on the right, in a program in which the transition condition from step 100 to 101 and the transition condition from step 101 to 102 are the same, step 100 becomes active in the previous scan, and when device Ⓐ has been switched ON in the present scan, there is transition to step 101 in the present scan. (Transition to step 102 will be from the next scan onward)



(3) Step transition processing means making the previous step inactive and the following step active if the transition condition is satisfied, based on the result of evaluation of the transition condition.

(4) Execution of the action program corresponding to the active step is carried out by switching the power rail ON, and executing the action program corresponding to the inactive step by switching the power rail OFF. At this time, as shown in the following diagram, the execution sequence is from top to bottom, and from left to right in branches.



The numerals in the diagram show the execution sequence of the action programs.

Points to note The following is a list of points to note when creating SFC programs.

(1) The capacity limits of SFC programs are set out in the following Tables. Be careful not to exceed these capacities.

- Overall Capacities (Maximum numbers which can be programmed in the T3)

Number of SFC main programs	64 (0-63)
Number of macro programs	128 (0-127)
Number of SFC steps	4096 (0-4095)
Number of SFC labels	1024 (0-1023)

- Capacities per SFC Main Program/Macro Program

Number of SFC steps	128
Number of instruction steps (SFC, actions and transition conditions total)	1024 steps*
Number of simultaneous branches	5
SFC edit screen capacity	128 lines by 5 columns

- Capacities per Action/Transition condition

Action program capacity	121steps*
Transition condition capacity	110 steps*

*) See 5.5 'List of instructions' for the required numbers of steps for SFC instructions and ladder diagram instructions.

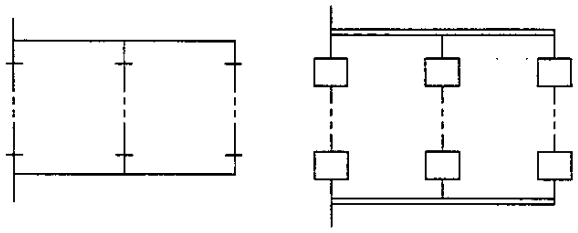
(2) The starting and re-setting of an SFC program is carried out by the SFC initialization instruction (SFC instruction/ladder diagram instruction). SFC initialization makes the steps in a designated area inactive and makes the initial step active. Therefore, the area of the steps designated by SFC initialization (the number of initialized steps) includes all the step numbers which are used in that SFC program (including macro programs as well). Take care that step numbers used in other SFC programs are not involved.

For instance, if the SFC initialization designation is 50 steps from step number 0 and step 50 is used in that SFC program, when SFC initialization is executed with step 50 in the active state, step 50 will remain active.

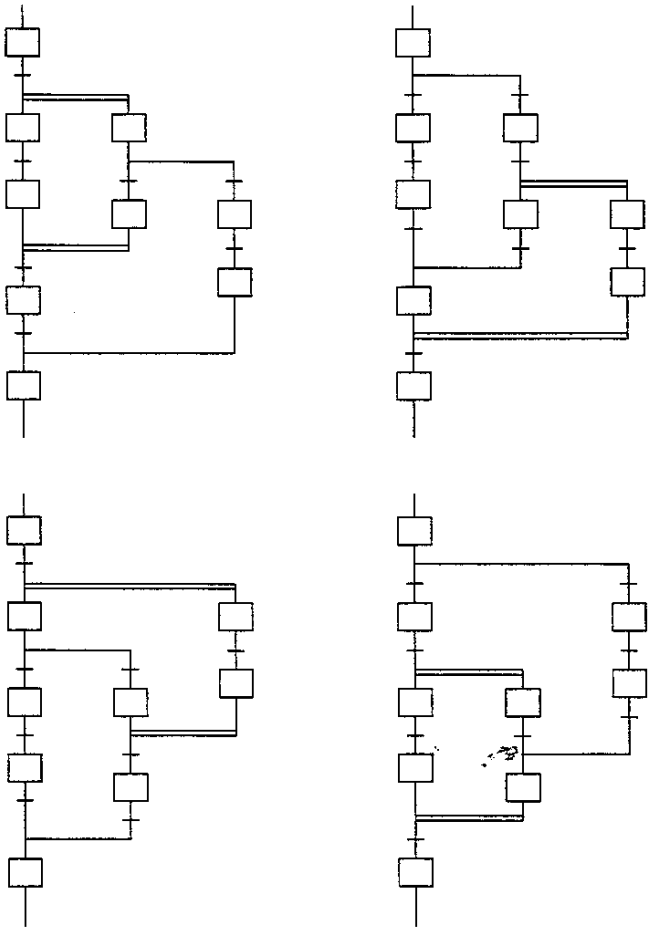
On the other hand, if the SFC initialization designation is 201 steps from step number 100 and step 300 is used in another SFC program, when SFC initialization is executed with step 300 in the active state, step 300 will become inactive without any condition.

(3) There is no limit to the step number sequence used in 1 SFC program (including macro programs). However, the initial step must be made the lowest step number in that sequence. (See (2) above)

(4) A sequence selection diverges above transitions, and converges below transitions. Also, a simultaneous sequence diverges above a steps and converges below a steps.



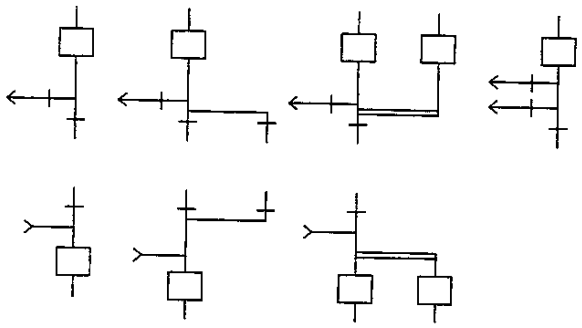
However, the divergence must end in a corresponding convergence. Therefore, programs such as the following are not allowed.



(5) The jump destination of a SFC jump may be either in the upward direction or in the downward direction, or it may be in another SFC program. Also, it is possible to jump to the outside from inside a branch.

Since a SFC jump can be very freely used in this way, take thorough precautions so that the SFC logic will not become abnormal (so that multiple unrelated steps in a series of SFC will not become active) through jumping.

A SFC jump is always positioned immediately after a step, Also, although basically a SFC label is positioned immediately after a transition, it is positioned between the convergence line and the step in the case of a sequence selection (convergence).



(6) The states (active/inactive) of SFC steps are not retained for power off. When starting-up, all become inactive.

(7) The output of an SFC step can be controlled by sandwiching the SFC program block by ladder diagram master control (MCS/MSR). When the input of MCS is OFF, the power rail of the action program corresponding to the active step also becomes OFF. However, in the state, step transition is carried out.

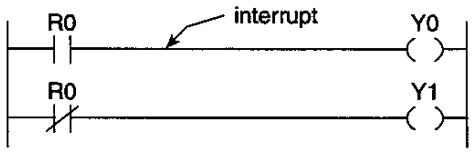
5.4 Programming precautions

The T3 supports multi-task function. When using this function, there is the possibility of the sub-program being interrupted by the main program or the interrupt program, and the main program being interrupted by the interrupt program. Precautionary notes arising from this are given below, and should be taken into account when creating programs.

(1) Avoid using the same sub-routine in the main program, the sub-programs and the interrupt programs. When the main program exection is interrupted during a sub-routine is being executed and the same sub-routine is executed in that state, the results after re-starting are sometimes not as expected.

(2) There is no classification of user data (register/device) by program type.Therefore, take thorough precautions that there is no erroneous mixed use between program types.

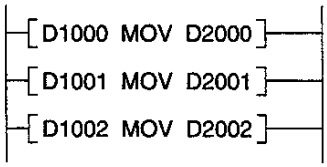
Example)



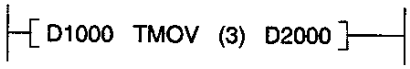
Interrupt occurs through the timing in the above diagram. And when the content of R0 is modified in the interrupt, the simultaneous ON (or the simultaneous OFF) of Y0 and Y1, which normally could not occur, happens.

(3) Try to execute the exchange of data between different program types by 1 instruction or by using the interrupt disable (DI) and the interrupt enable (EI) instructions. Otherwise, the same thing as in (2) above may happen.

Example) Composition of the main program when transferring the three data, D1000, D1001 and D1002, from the interrupt program to the main program.



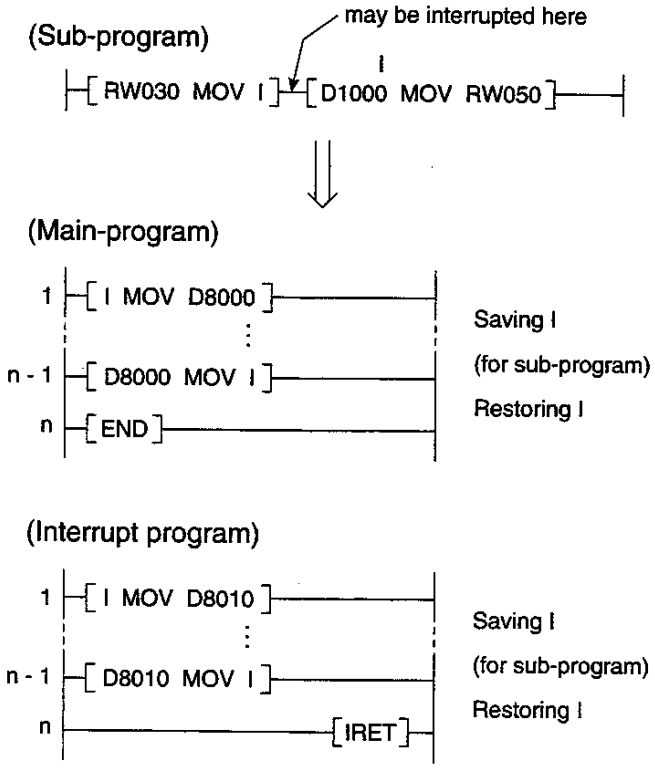
In the above program, when an interrupt occurs between instructions, synchronization between D2000, D2001 and D2002 cannot be guaranteed. In this case, make 1 instruction by using the table transfer instruction, as follows.



Or sandwich these instructions by DI and EI instructions.

(4) If the same index register is used in different program types, the data of the index register should be saved and restored as follows.

Example)



With respect to the main program, the data of index registers are saved when interrupt occurs and restored when operation returns to main program automatically. However, because of this, even if an index register is used only in an interrupt program, the data continuity of the index register between interrupt intervals is not kept. In such case, use another register to store index value substitute the value into an index register in the interrupt program.

5.5 List of instructions

An instruction list is given in the sequence of ladder diagram instructions and SFC instructions on the next page and thereafter.

The groups in the list correspond to the group classifications of function instructions used in the programmer (T-PDS). (Except for SFC).

The required numbers of steps signify the size of memory required for storing these instructions. The showing of the required number of steps by a range such as 4-7, is because the number of steps changes due to the following conditions, even for the same instruction.

- When using digit designation, there is an increase of 1 step per 1 operand.
- When a constant is used in a double-length operand, there is an increase of 1 step.
- When executing index modification in a constant, there is an increase of 1 step.

The minimum execution time figure shows normal case value, i.e. when no index modification, no digist designation and normal registers are used for each operand.

The maximum execution time figure shows worst case value, i.e. when direct input/output registers (IW/OW) are used for each operand, etc.

NOTE

Here, an overview of each instruction is given. See the instruction set manual in a separate volume for details.

Ladder Diagram Instructions (Sequence Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Sequence instructions		No contact	(A) ⊣ ⊢	NO contact of device (A) (contact normally open)	1	—	0.15	7.45
		NC contact	(A) ⊣/⊢	NC contact of device (A) (contact normally closed)	1	—	0.15	7.45
		Transitional contact (rising)	⊣↑⊢	Switches output ON only when input in the previous scan is OFF and the input in this scan is ON.	1	—	0.60	—
		Transitional contact (falling)	⊣↓⊢	Switches output ON only when input in the previous scan is ON and input in this scan is OFF.	1	—	0.60	—
		Coil	(A) ⊣( )⊢	Switches device (A) ON when input is ON.	1	—	0.30	7.45
		Forced coil	(A) ⊣*( )⊢	Retains state of device (A) regardless of whether input is ON or OFF.	1	—	0.15	—
		Inverter	(A) ⊣I⊢	Inverts the input state	1	—	0.15	—
		Invert coil	(A) ⊣(I)⊢	Inverts the input state and stores in device (A).	1	—	0.30	7.45
		Positive Transition-sensing contact	(A) ⊣P⊢	Turns output ON for 1 scan when input is ON and device (A) is changed from OFF to ON.	1	—	0.60	7.75
		Negative Transition-sensing contact	(A) ⊣N⊢	Turns output ON for 1 scan when input is ON and device (A) is changed from ON to OFF.	1	—	0.60	7.75
		Positive Transition-sensing coil	(A) ⊣(P)⊢	Turns device (A) ON for 1 scan when input is changed from OFF to ON.	1	—	0.60	7.75
		Negative Transition-sensing coil	(A) ⊣(N)⊢	Turns device (A) ON for 1 scan when input is changed from ON to OFF.	1	—	0.60	7.75
		Jump control set	⊣[JCS]⊢	Carries out high-speed skipping on instructions between JCS and JCR when input is ON.	1	—	0.15	—
		Jump control reset	⊢[JCR]⊢		1	—	0.15	—
		End	⊢[END]⊢	Indicates end of main program and sub-program.	1	—	—	—

Ladder Diagram Instructions (Sequence Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Sequence instructios		ON delay timer	⊣[(A) TON (B)]⊢	Turns output ON when set period specified by (A) has elapsed since input came ON. (B) is timer register.	2	—	0.30	8.55
		OFF delay timer	⊣[(A) TOF (B)]⊢	Turns output OFF when set period specified by (A) has elapsed since input went OFF. (B) is timer register.	2	—	0.30	8.55
		Single shot timer	⊣[(A) SS (B)]⊢	Turns output ON only for the set period, specified by (A), starting when input comes ON. (B) is timer register.	2	—	0.30	8.55
		Counter	⊣[C CNT Q E (A) (B)]⊢	When enable input (E) is ON, counts the number of times the count input (C) has come ON. When count value becomes equal to set value specified by (A), turns output (Q),ON. (B) is counter register.	2	—	0.30	17.3
		Master control set	⊣[MCS]⊣	Turns ON power rail between MCS and MCR when MCS input is ON.	1	—	0.15	—
		Master control reset	⊢[MCR]⊣		1	—	0.15	—
	134	Master control set (with nesting number)	⊣[MCS n]⊣	Turns ON power rail to corresponding MCR when MCS input is ON. n is a nesting number. (1 - 7).	2	—	0.30	4.95
	135	Master control reset (with nesting number)	⊢[MCR n]⊣		2	—	0.30	4.7
	148	Timer trigger	⊣[TRG (A)]⊢	When input is changed from OFF to ON, clears timer register specified by (A) and activates timer.	2	0.30	6.5	8.55

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Transfer instructions	18	Data transfer	—[(A) MOV (B)]—	Transfers contents of (A) to (B).	3~5	0.45	0.9	112.8
	19	Double-length data transfer	—[(A)+1 · (A) DMOV (B)+1 · (B)]—	Transfers contents of (A)+1 and (A) to (B)+1 and (B).	3~6	0.45	7.88	149.3
	20	Invert and transfer	—[(A) NOT (B)]—	Transfers the bit-reversed data comprising the contents of (A) to (B).	3~5	0.45	6.75	113.1
	21	Double-length invert and transfer	—[(A)+1 · (A) DNOT (B)+1 · (B)]—	Transfers the bit-reversed data comprising the contents of (A)+1 and (A) to (B) +1 and (B).	3~6	0.45	9.45	149.8
	22	Data exchange	—[(A) XCHG (B)]—	Exchanges the contents of (A) with the contents of (B).	3~5	0.45	13.5	184.2
	23	Double-length data exchange	—[(A)+1 · (A) DXCH (B)+1 · (B)]—	Exchanges the contents of (A)+1 · (A) with the contents of (B)+1 · (B)	3~5	0.45	16.9	216.5
	24	Table initialization	—[(A) TINZ (n) (B)]—	Initializes the contents of the table of size n, headed by (B), by the contents of (A).	4~6	0.6	49.1+0.9n	71.6+16.5n
	25	Table transfer	—[(A) TMOV (n) (B)]—	Transfers the contents of the table of size n, headed by (A), to the table headed by (B).	4~6	0.6	77.2+1.13n	235.3+34n
	26	Table invert and transfer	—[(A) TNOT (n) (B)]—	Transfers the bit-reversed data comprising the contents of the table of size n headed by (A) to the table headed by (B).	4~6	0.6	77.2+1.13n	235.9+34.3n
Arithmetic operations	27	Addition	—[(A) + (B)→(C)]—	Adds the contents of (B) to the contents of (A), and stores the result in (C).	4~7	0.6	2.25	163.1
	28	Subtraction	—[(A) - (B)→(C)]—	Subtracts the contents of (B) from the contents of (A), and stores the result in (C).	4~7	0.6	2.25	163.1
	29	Multiplication	—[(A) * (B) → (C)+1 · (C)]—	Multiplies the contents of (A) by the contents of (B) and stores the result in (C) + 1 · (C).	4~7	0.6	6.53	183.1
	30	Division	—[(A) / (B)→ (C)]—	Divides the contents of (A) by the contents of (B), stores the quotient in (C), and the remainder in (C)+1.	4~7	0.6	11.5	190.5
	31	Double-length addition	—[(A)+1 · (A) D+ (B)+1 · (B) → (C)+1 · (C)]—	Adds the contents of (B)+1 · (B) to the contents of (A)+1 · (A), and stores the result in (C)+1 · (C).	4~9	0.6	12.2	219.8
	32	Double-length subtraction	—[(A)+1 · (A) D- (B)+1 · (B) → (C)+1 · (C)]—	Subtracts the contents of (B)+1·(B) from the contents of (A)+1 · (A), and stores the result in (C)+1 · (C).	4~9	0.6	12.2	219.8

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Arithmetic operations	33	Double-length multiplication	—[(A)+1·(A) D * (B)+1·(B) → (C)+1·(C)]—	Multiplies the contents of (A)+1 · (A) by the contents of (B)+1 · (B). and stores the result in (C)+3 · (C)+2 · (C)+1 · (C).	4~9	0.6	49.3	254.0
	34	Double-length division	—[(A)+1·(A) D/ (B)+1·(B) → (C)+1·(C)]—	Divides the contents of (A)+1 · (A) by the contents of (B)+1 · (B), and stores the quotient in (C)+1 · (C) and the remainder in (C)+3 · (C)+2.	4~9	0.6	27.2	299.5
	35	Addition with carry	—[(A) +C (B) → (C)]—	Adds the contents of the carry flag and the contents of (B) to the contents of (A), and stores the result in (C). The carry flag changes according to the operation result.	4~7	0.6	11.9	164.9
	36	Subtraction with carry	—[(A) -C (B) → (C)]—	Subtracts the contents of (B) and the contents of the carry flag from the contents of (A), and stores the result in (C). The carry flag changes according to the operation result.	4~7	0.6	11.9	164.9
	37	Double-length addition with carry	—[(A)+1·(A) D+C (B)+1·(B)→(C)+1·(C)]—	Adds the contents of the carry flag to the contents of (A)+1 · (A) and the contents of (B)+1 · (B), and stores the result in (C)+1 · (C). The carry flag changes according to the operation result.	4~9	0.6	14.0	221.6
	38	Double-length subtraction with carry	—[(A)+1·(A) D-C (B)+1·(B)→(C)+1·(C)]—	Subtracts the contents of (B)+1 · (B) plus the contents of the carry flag from the contents of (A)+1 · (A), and stores the result in (C)+1 · (C). The carry flag changes according to the operation result.	4~9	0.6	14.0	221.6
	39	Unsigned multiplication	—[(A) U * (B) → (C)+1·(C)]—	Multiplies the contents of (A) by the contents of (B), and stores the result in (C)+1 · (C) (unsigned integer calculation).	4~7	0.6	16.0	184.5
	40	Unsigned division	—[(A) U/ (B) → (C)]—	Divides the contents of (A) by the contents of (B), and stores the quotient in (C), and the remainder in (C)+1 (unsigned integer operation).	4~7	0.6	16.9	185.4
	41	Unsigned double/single division	—[(A)+1·(A) DIV (B) → (C)]—	Divides the contents of (A)+1 · (A) by the contents of (B), stores the quotient in (C), and the remainder in (C)+1 (unsigned integer operation).	4~8	0.6	23.9	206.5
	43	Increment	—[+1 (A)]—	Increments the contents of (A) by 1.	2~3	0.3	6.98	112.4
	44	Double-length increment	—[D+1 (A)+1·(A)]—	Increments the contents of (A)+1 · (A) by 1.	2~3	0.3	9.0	149.1
	45	Decrement	—[-1 (A)]—	Decrements the contents of (A) by 1.	2~3	0.3	6.98	112.4
	46	Double-length decrement	—[D-1 (A)+1·(A)]—	Decrements the contents of (A)+1 · (A) by 1.	2~3	0.3	9.0	149.1

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Arithmetic operations	208	Floating point addition	—[(A)+1·(A) F+ (B)+1·(B) → (C)+1·(C)]—	Adds the floating point data of (A)+1 · (A) and (B)+1 · (B), and stores the result in (C)+1 · (C).	4	0.6	33.3	199.0
	209	Floating point subtraction	—[(A)+1·(A) F- (B)+1·(B) → (C)+1·(C)]—	Subtracts the floating point data of (B)+1 · (B) from (A)+1 · (A), and stores the result in (C)+1 · (C).	4	0.6	34.2	199.9
	210	Floating point multiplication	—[(A)+1·(A) F* (B)+1·(B) → (C)+1·(C)]—	Multiplies the floating point data of (A)+1 · (A) by (B)+1 · (B), and stores the result in (C)+1 · (C).	4	0.6	54.9	220.5
	211	Floating point division	—[(A)+1·(A) F/ (B)+1·(B) → (C)+1·(C)]—	Divides the floating point data of (A)+1 · (A) by (B)+1 · (B), and stores the result in (C)+1 · (C).	4	0.6	35.1	200.8
Logical operations	48	AND	—[(A) AND (B) → (C)]—	Finds the logical AND of (A) and (B) and stores it in (C).	4~7	0.6	9.68	162.6
	49	Duuble-length AND	—[(A)+1·(A)DAND (B) +1 · (B)→(C)+1·(C)]—	Finds the logical AND of (A)+1 · (A) and (B)+1 · (B) and stores it in (C)+1 · (C).	4~9	0.6	11.7	219.4
	50	OR	—[(A) OR (B) →(C)]—	Finds the logical OR of (A) and (B) and stores in (C).	4~7	0.6	9.68	162.6
	51	Double-length OR	—[(A)+1·(A) DOR (B)+1·(B)→(C)+1·(C)]—	Finds the logical OR of (A)+1 · (A) and (B)+1 · (B) and stores it in (C)+1 · (C).	4~9	0.6	11.7	219.4
	52	Exclusive OR	—[(A) EOR (B)→(C)]—	Finds the exclusive logical OR of (A) and (B) and stores it in (C).	4~7	0.6	9.68	162.6
	53	Double-length exclusive OR	—[(A)+1·(A) DEOR (B)+1·(B)→(C)+1·(C)]—	Finds the exclusive logical OR of (A)+1 · (A) and (B)+1 · (B) and stores it in (C)+1 · (C).	4~9	0.6	11.7	219.4
	54	Not exclusive OR	—[(A) ENR (B)→(C)]—	Finds the negative exclusive OR of (A) and (B) and stores it in (C).	4~7	0.6	9.68	162.6
	55	Double-length Not exclusive OR	—[(A)+1·(A) DENR (B)+1·(B)→(C)+1·(C)]—	Finds the negative exclusive OR of (A)+1 · (A) and (B)+1 · (B) and stores it in (C)+1 · (C).	4~9	0.6	11.7	219.4

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Logical operations	57	Table AND	─[(A) TAND (n) (B)→(C)]─	Finds the logical AND of the table of size n headed by (A) and the table of size n headed by (B), and stores it in the location headed by (C).	5	0.75	46.1+1.8n	148.9+21.9n
	58	Table OR	─[(A) TOR (n) (B)→(C)]─	Finds the logical OR of the table of size n headed by (A) and the table of size n headed by (B), and stores it in the location headed by (C).	5	0.75	46.1+1.8n	148.9+21.9n
	59	Table exclusive OR	─[(A) TEOR (n) (B)→(C)]─	Finds the exclusive OR of the table of size n headed by (A) and the table of size n headed by (B), and stores it in the location headed by (C).	5	0.75	46.1+1.8n	148.9+21.9n
	60	Table Not exclusive OR	─[(A) TENR (n) (B)→(C)]─	Finds the NOT exclusive OR of the table of size n headed by (A) and the table of size n headed by (B) and stores it in the location headed by (C).	5	0.75	46.1+1.8n	148.9+21.9n
	64	Test	─[(A) TEST (B)]─	Turns the output ON if the logical AND of (A) and (B) is other than 0.	3~5	0.45	6.98	100.2
	65	Double-length test	─[(A)+1 · (A) DTST (B)+1 · (B)]─	Turns the output ON if the logical AND of (A)+1 · (A) and (B)+1 · (B) is other than 0.	3~7	0.45	8.33	140.8
	66	Bit file bit test	─[(A) TTST (n) (B)]─	Decides the ON/OFF state of the (A)th bit of the bit table size n headed by (B).	4~5	0.6	27.0	38.3
Shifts	68	1 bit shift right	─[SHR 1 (A)]─	Shifts the data in (A) 1 bit to the right (LSB direction) and stores the result in (A). The carry flag chages according to the result.	2~3	0.3	8.55	114.9
	69	1 bit shift left	─[SHL 1 (A)]─	Shifts the data in (A) 1 bit to the left (MSB direction) and stores the result in (A). The carry flag chnges according to the result.	2~3	0.3	11.3	117.6
	70	n bit shift right	─[(A) SHR n→(B)]─	Shifts the data in (A) n bits to the right (LSB direction) and stores the result in (B). The carry flag changes according to the result.	4~6	0.6	9.68+0.68n	116.0+0.68n
	71	n bit shift left	─[(A) SHL n→(B)]─	Shifts the data in (A) n bits to the left (MSB direction) and stores the result in (B). The carry flag changes according to the result.	4~6	0.6	10.8+0.68n	119.1+0.68n

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Shift	72	m bit file n bits shift right	(A) TSHR (m)→(B)	When (B) is a register: Takes the m-word table headed by (B), and shifts it to the right (low address direction) by the number of words indicated by (A). When (B) is a device: Takes the m-bit file headed by (B), and shifts it to the right (LSB direction) by the number of bits indicated by (A). The carry flag chnges according to the result.	4~5	0.6	26.8+ 1.13m+ 0.23n	97.6+ 0.78m+ 0.44n
	73	m bit file n bits shift left	(A) TSHL (m)→(B)	When (B) is a register: Takes the m-word table headed by (B), and shifts it to the left (high address direction) by the number of words indicated by (A). When (B) is a device: Takes the m-bit file headed by (B), and shifts it to the left (MSB direction) by the number of bits indicated by (A). The carry flag chnges according to the result.	4~5	0.6	27.7+ 1.13m+ 0.23n	90.0+ 0.78m+ 0.44n
	74	Shift register	D SR Q S (n) E (A)	If the enable input (E) is ON, then when the shift input (S) comes ON, the instruction takes the contents of the n devices headed by the device (A) and shifts them 1 bit to the left. The carry flag changes according to the result.	3	0.45	35.8+5.18n	55.1+5.18n
	75	Bidirectional shift register	D DSR Q S (n) E L (A)	If the enable input (E) is ON, then when the shift input (S) comes ON, the instruction takes the contents of the n devices headed by the device (A) and shifts them 1 bit to the left or to the right (the shift direction depends on the state of the direction input (L)). The carry flag chnges according to the result.	3	0.45	35.8+6.75n	65.1+6.75n
	76	Device shift	SFT (A)	Takes the contents of the device ((A)-1) which immediately precedes the device (A), stores it in (A), and sets (A)-1 to 0	2	0.3	30.2	75.6

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Rotate	78	1 bit rotate right	⊣[RTR 1 (A)]⊢	Rotates the data in (A) 1 bit to the right (LSB direction). The carry flag changes according to the result.	2~3	0.3	9.23	117.8
	79	1 bit rotate left	⊣[RTL 1 (A)]⊢	Rotates the data in (A) 1 bit to the left (MSB direction). The carry flag chnges according to the result.	2~3	0.3	8.78	117.6
	80	n bits rotate right	⊣[(A) RTR n→(B)]⊢	Rotates the data in (A) n bits to the right (LSB direction). The carry flag changes according to the result.	4~6	0.6	11.5+ 0.23n	117.8+ 1.8n
	81	n bits rotate left	⊣[(A) RTL n→(B)]⊢	Rotates the data in (A) n bits to the left (MSB direction). The carry flag changes according to the result.	4~6	0.6	10.8+ 0.23n	117.6+ 1.8n
	82	m bit file n bits rotate right	⊣[(A) TRTR (m) (B)]⊢	When (B) is a register: Takes the table of m words, headed by (B), and rotates it to the right (low address direction) by the number of words specified by (A). When (B) is a device: Takes the bit file of m bits, headed by (B), and rotates it to the right (LSB direction) by the number of bits specified by (A). The carry flag changes according to the result.	4~5	0.6	30.6+ 2.25m	103.0+ 0.84m+ 0.53n
	83	m bit file n bits rotate left	⊣[(A) TRTL (m) (B)]⊢	When (B) is a register: Takes the table of m words, headed by (B), and rotates it to the left (high address direction) by the number of words specified by (A). When (B) is a device: Takes the bit file of m bits, headed by (B), and rotates it to the left (MSB direction) by the number of bits specified by (A). The carry flag changes according to the result.	4~5	0.6	30.6+ 2.25m	108.0+ 0.84m+ 0.6n
	84	1 bit rotate right with carry	⊣[RRC 1 (A)]⊢	Rotates the data in (A) 1 bit to the right (LSB direction) including the carry flag. The carry flag changes according to the result.	2~3	0.3	9.9	117.6

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Rotate	85	1 bit rotate left with carry	—[RLC1 (A)]—	Rotates the data in (A)1 bit to the left(MSB direction) including the carry flag. The carry flag changes according to the result.	2~3	0.3	9.45	117.6
	86	n bits rotate right with carry	—[(A) RRCn → (B)]—	Rotates the data in (A)n bit to the left(LSB direction) including the carry flag, and stores the result in (B). The carry flag changes according to the result.	4~6	0.6	9.9+ 2.03n	117.6+ 2.02n
	87	n bits rotate left with carry	—[(A) RLC n → (B)]—	Rotates the data in (A)n bit to the left(MSB direction) including the carry flag, and stores the result in (B). The carry flag changes according to the result.	4~6	0.6	11.3+ 1.8n	118.9+ 1.8n

	88	m bit file n bits rotate right with carry	—[(A) TRRC (m) (B)]—	If (B) is a register:
				Takes the table of m words headed by (B) and rotates it to the right (low address direction) by the number of words indicated by (A). (Same as register specification in FUN82.)
				--- 	
				If (B) is a device:
				Takes the bit file of m bits headed by (B), including the carry flag, and rotates it to the right (LSB direction) by the number of bits indicated by (A). The carry flag changes according to the result.
					4~5	0.6	30.6+ 2.25m	116.7+ 0.77m+ 0.45n

	89	m bit file n bits rotate left with carry	—[(A) TRLC (m) (B)]—	If (B) is a register:
				Takes the table of m words headed by (B) and rotates it to the right (high address direction) by the number of words indicated by (A). (Same as register specification in FUN83.)
				--- 	
				If (B) is a device:
				Takes the bit file of m bits headed by (B), including the carry flag, and rotates it to the right (MSB direction) by the number of bits indicated by (A). The carry flag changes according to the result.
					4~5	0.6	30.6+ 2.25m	107.5+ 0.84m+ 0.53n

	90	Multiplexer	—[(A) MPX (n) (B) → (C)]—	Takes the contents of the (B)th register in the table of size n headed by the register(A), and stores them in the register(C).	5~6	0.75	29.3	68.9

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Rotate	91	Demultiplexer	—[(A) DPX (n) (B) → (C)]—	Stores the contents of the register(A) in the (B)th register of the table of size n headed by the register(C).	5~6	0.75	25.9	57.3
	92	Table→bit transfer	—[(A) TBM (n) (B) → (C)]—	Takes the (B)th bit from the head of the table of size n words headed by the register(A) and stores it in the device(C).	5~6	0.75	36.1	73.9
	93	Bit→table transfer	—[(A) BTM (n) (B) → (C)]—	Takes the contents of the device(A) and stores them in the (B)th bit of the table of size n headed by the register(C).	5~6	0.75	32.7	60.7

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Compare	95	Bit file comparison	—[(A) TCMP (n) (B) → (C)]—	Conpares the register tables starting from (A) and (B), and stores the non-matching bits in (C).	5	0.75	67.6+ 2.48n	—
	96	Greater than	—[(A) > (B)]—	Turns output ON if (A)>(B) (inerger comparison).	3~5	0.45	6.75	100
	97	Greater than or equal	—[(A) >= (B)]—	Turns output ON if (A)≥(B) (inerger comparison).	3~5	0.45	6.98	100.2
	98	Equal	—[(A) = (B)]—	Turns output ON if (A)=(B) (inerger comparison).	3~5	0.45	6.98	100.2
	99	Not equal	—[(A) <> (B)]—	Turns output ON if (A)≠(B) (inerger comparison).	3~5	0.45	6.98	100.2
	100	Less than	—[(A) < (B)]—	Turns output ON if (A)<(B) (inerger comparison).	3~5	0.45	6.98	100.2
	101	Less than or equal	—[(A) <= (B)]—	Turns output ON if (A)≤(B) (inerger comparison).	3~5	0.45	6.98	100.2
	102	Double-length greater than	—[(A)+1 · (A) D> (B)+1 · (B)]—	Turns output ON if (A)+1 · (A)>(B)+1 · (B) (double-length integer comparison).	3~7	0.45	8.33	140.8
	103	Double-length greater than or equal	—[(A)+1 · (A) D>= (B)+1 · (B)]—	Turns output ON if (A)+1 · (A)≥(B)+1 · (B) (double-length integer comparison).	3~7	0.45	8.1	140.6
	104	Double-length equal	—[(A)+1 · (A) D= (B)+1 · (B)]—	Turns output ON if (A)+1 · (A)=(B)+1 · (B) (double-length integer comparison).	3~7	0.45	8.33	140.8
	105	Double-length not equal	—[(A)+1 · (A) D<> (B)+1 · (B)]—	Turns output ON if (A)+1 · (A)≠(B)+1 · (B) (double-length integer comparison).	3~7	0.45	8.33	140.8
	106	Double-length less than	—[(A)+1 · (A) D< (B)+1 · (B)]—	Turns output ON if (A)+1 · (A)<(B)+1 · (B) (double-length integer comparison).	3~7	0.45	8.33	140.8
	107	Double-length less than or equal	—[(A)+1 · (A) D<= (B)+1 · (B)]—	Turns output ON if (A)+1 · (A)≤(B)+1 · (B) (double-length integer comparison).	3~7	0.45	8.1	140.6

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Compare	108	Unsigned greater than	—[(A) U> (B)]—	Turns output ON if (A)>(B) (unsigned integer comparison).	3~5	0.45	6.98	100.2
	109	Unsigned greater than or equal	—[(A) U>= (B)]—	Turns output ON if (A)≥(B) (unsigned integer comparison).	3~5	0.45	6.98	100.2
	110	Unsigned equal	—[(A) U= (B)]—	Turns output ON if (A)=(B) (unsigned integer comparison).	3~5	0.45	6.98	100.2
	111	Unsigned not equal	—[(A) U<> (B)]—	Turns output ON if (A)≠(B) (unsigned integer comparison).	3~5	0.45	6.98	100.2
	112	Unsigned less than	—[(A) U< (B)]—	Turns output ON if (A)<(B) (unsigned integer comparison).	3~5	0.45	6.98	100.2
	113	Unsigned less than or equal	—[(A) U<= (B)]—	Turns output ON if (A)≤(B) (unsigned integer comparison).	3~5	0.45	6.98	100.2
	212	Floating point greater than	—[(A)+1·(A) F> (B)+1·(B)]—	Turns output ON if (A)+1·(A)>(B)+1·(B) (floating point data comparison).	3	0.45	20.5	124.7
	213	Floating point greater than or equal	—[(A)+1·(A) F>= (B)+1·(B)]—	Turns output ON if (A)+1·(A)≥(B)+1·(B) (floating point data comparison).	3	0.45	20.5	124.7
	214	Floating point equal	—[(A)+1·(A) F= (B)+1·(B)]—	Turns output ON if (A)+1·(A)=(B)+1·(B) (floating point data comparison).	3	0.45	16.7	121.9
	215	Floating point not equal	—[(A)+1·(A) F<> (B)+1·(B)]—	Turns output ON if (A)+1·(A)≠(B)+1·(B) (floating point data comparison).	3	0.45	16.7	121.9
	216	Floating point less than	—[(A)+1·(A) F< (B)+1·(B)]—	Turns output ON if (A)+1·(A)<(B)+1·(B) (floating point data comparison).	3	0.45	20.5	124.7
	217	Floating point less than or equal	—[(A)+1·(A) F<= (B)+1·(B)]—	Turns output ON if (A)+1·(A)≤(B)+1·(B) (floating point data comparison).	3	0.45	20.5	124.7

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum

Special data processing	114	Set device/register	—[SET (A)]—	If (A) is a device:
				Sets device(A) to ON.
				If (A) is a register:
				Stores HFFFF in register(A).
					2~3	0.3	6.53
							4.28
								30.4
								27.0

	115	Rerest device/register	—[RST (A)]—	if (A) is a device:
				Sets device(A) to OFF.
				If (A) is a register:
				Stores 0 in register(A).
					2~3	0.3	6.53
							4.28
								30.4
								27.0

	116	Table bit set	—[(A) TSET (n) (B)]—	From the bit file of n words, headed by the register(B), the instruction takes the bit in the location indicated by (A) and sets it to ON.	4~5	0.6	31.7	53.0
	117	Table it reset	—[(A) TRST (n) (B)]—	From the bit file of n words, headed by the register(B), the instruction takes the bit in the position indicated by (A) and resets it to OFF.	4~5	0.6	31.1	53.0
	118	Set carry	—[SETC]—	Sets the carry flag.	1	0.15	2.48	—
	119	Reset catty	—[RSTC]—	Resets the carry flag.	1	0.15	2.48	—
	120	Encode	—[(A) ENC (n) (B)]—	In the bit file of size 2^n bits headed by (A), the instruction stores the uppermost ON bit position in register(B).	3~4	0.45	29.0	1212.9
	121	Decode	—[(A) DEC (n) (B)]—	Takes the bit file of size 2^n bits headed by (B), sets the bit position indicated by the lower n bits of register(A) to ON, and sets all the rest to OFF.	3~4	0.45	30.4	1861.1
	122	Bit count	—[(A) BC (B)]—	Counts the number of ON bits in the data in (A) and stores the result in (B).	3~5	0.45	25.2	131.5
	123	Double-length bit count	—[(A) DBC (B)]—	Counts the number of ON bits in the double-length data in (A)+1 · (A) and stores the result in (B)+1 · (B).	3~6	0.45	43.9	169.8

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Special data processing	124	Data search	—[(A) SCH (n) (B) → (C)]—	Searches through data table of n words headed by (B) for data matching the contents of (A). Stores the number of matches in (C), and stores the lowest register address of the matching registers in (C)+1.	5~6	0.75	23.6+ 2.7n	96.0+ 12n
	125	Push	—[(A) PUSH (n) (B) → (C)]—	Pushes the data in (A) into the table of n words headed by (C), and increments the value of (B) by 1.	5~6	0.75	15.8	104.9+ 1.13n
	126	Pop last	—[(A) POPL (n) (B) → (C)]—	Takes out the data pushed in last to the table of n words headed by (A) and stores it in (C). Also decrements the value of (B) by 1.	5~6	0.75	16.4	106.7+ 1.34n
	127	Pop first	—[(A) POPF (n) (B) → (C)]—	Takes out from the table of n words headed by (A) the data which was pushed in first, and stores it in (C). Also decrements the value of (B) by 1.	5~6	0.75	16.8	109.1
	147	Flip-flop	—[S F/F Q —[R (A)]	When the set input (S) is ON, the instruction sets the device (A) to ON; when the reset input (R) is ON, it resets the device (A) to OFF. (Reset takes priority)	2	0.3	8.1	22.7
	149	Up-down counter	—[U U/D Q —C —[E (A)]	If the enable input (E) is ON, the instruction counts the number of times the count input (C) has come ON and stores it in the counter register(A). The selection of the count direction (increment/decrement) is made according to the state of the up/down selection input (U) (see below). ON:UP count(increment) OFF:DOWN count(decrement)	2	0.3	4.28	6.08

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary		Number of steps required	Execution time required (μs)		
							Not executed	Minimum	Maximum
Program control	128	Subroutine call	⊢[CALL N. nn]⊢	If the input is ON, the instruction calls the subroutine for the subroutine number nn.		2~3	0.3	0.45	142
	129	Subroutine return	⊢[RET]⊢	Indicates the end of the subroutine.		1	—	6.98	—
	130	Conditional jump	-[JUMP N. nn]-	If the input is ON, jumps directly to the label for the label number nn.		2~3	0.3	0.45	108
	136	Jump label	⊢[LBL (nn)]⊢	Indicates the jump destination for the conditional jump.		2	—	1.35	—
	132	FOR-NEXT loop (FOR)	-[FOR n]⊢	Executes the section from FOR to NEXT repeatedly the number of times specifide by n.		2	0.3	6.98	—
	133	FOR-NEXT loop (NEXT)	-[NEXT]⊢			1	0.15	4.95	94
	137	Subroutine entry	⊢[SUBR (nn)]⊢	Indedates the entrance to the subroutine (number nn).		2	—	1.35	—
	138	Stop	-[STOP]-	Stops the program execution (to HALT).		1	0.15	—	—
	140	Enable interrupt	-[EI]-	Enables execution of the interrupt program.		1	0.15	51.0	—
	141	Disable interrupt	-[DI]-	Disables execution of the interrupt program.		1	0.15	53.0	—
	142	Interrupt program end	⊢[IRET]⊢	Indicates the end of the interrupt program.		1	—	—	—
	143	Watchdog timer reset	-[WDT n]-	Extends the scan time over detection time.		2	0.3	32.0	—
	144	Step sequence initialize	-[STIZ (n) (A)]-	Turns OFF the n devices headed by device (A), and turns (A) ON (activation of step sequence).	These comprise one step sequence.	3	0.45	20.7	259
	145	Step sequence input	-[(A)]-	Turns output ON when input is ON and device (A) is ON.		2	0.3	9.0	—
	146	Step sequence output	-[(A)]⊢	When input is ON, the instruction turns OFF the devices with step sequence input instructions on the same rung, and truns device (A) ON.		2	0.3	8.1	—
	241	SFC initialize	-[SFIZ (n) (A)]-	When input is changed fron OFF to ON, the instruction resets the n steps from the SFC step (A), and activates step (A)(activation of SFC).		3	0.45	9.23+0.9× INT(n/16-1)	—

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
RAS	150	Diagnostic display	⊣[DIAG (A) (B)]⊢	When input has changed from OFF to ON, the instruction records the error code indicated by (A) in the special register, and turns ON the corresponding annunciator relay. The error messages (max 12 characters) recorded in the register tables headed by (B) can be monitored on the peripheral devices.	3~4	0.45	24.3	36.5
	151	Diagnostic display reset	⊣[DIAR (A)]⊢	Erases the error code(A) from the error code list recorded by the diagnostic display instruction (FUN150) and from the annunciator relay.	2~3	0.3	16.0+3.83n	34.9+3.83n
	152	Status latch set	⊣[STLS]⊢	Takes the devices/registers(max 32) set by the programmer and stores them in the latch area.	1	0.15	944	1960
	153	Status latch reset	⊣[STLR]⊢	Cancels the state of the status latch.	1	0.15	126	544
	154	Set Calendar	⊣[(A) CLND]⊢	Takes the 6 words of data headed by the register(A) and sets them in the calendar LSI (date and time setting).	2	0.3	856.0	—
	155	Calendar operation	⊣[(A) CLDS (B)]⊢	Subtracts the 6 words of date and time data headed by (A), from the current date and time, and stores the result in the 6 words starting with (B).	3	0.45	1502.0	—
	158	Drum sequencer	⊣[(A) DRUM (n) (B) → (C) (m)]⊢	Compares the count value (B) With the count value setting table ((A)+2n onwards), then decides the step number and stores it in (B)+1. Using the data output pattern table(A), the instruction looks up the output pattern corresponding to this step number and outputs it to the bit table (C).	6	—	At initialize: 54.1 Executing: 60.1	—
	159	Cam sequencer	⊣[(A) CAM (n) (B) → (C)]⊢	Compares the register(B) with the activation and deactivation setting value for table (A), and carries out ON/OFF control on the corresponding devices.	5	0.75	20.9+ 11.3n	—
Function	160	Upper limit	⊣[(A) UL (B) → (C)]⊢	Applies an upper limit to the contents of (A) using the value of (B), and stores the results in (C).	4~7	0.6	10.58	163.5

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Function	161	Lower limit	(A) LL (B) → (C)	Applies a lower limit to the contents of (A), using the value of (B), and stores the results in (C).	4~7	0.6	10.58	163.5
	162	Maximum value	(A) MAX (n) (B)	Searches the n-word data table headed by (A) for the maximum value, stores the maximum value in (B), and stores the pointer with the maximum value in (B)+1.	4	0.6	20.3	83.8+11.7n
	163	Minimum value	(A) MIN (n) (B)	Searches the n-word data table headed by (A) for the minimum value, stores the minimum value in (A), and stores the pointer with the minimum value in (B)+1.	4	0.6	20.3	83.8+11.7n
	164	Average value	(A) AVE (n) (B)	Calculates the average value for the n-word data table headed by (A), and stores it in (B).	4	0.6	28.6	22.0+12n
	165	Function generatior	(A) FG (n) (B) → (C)	Using the function defined by the 2x n parameters headed by (B), finds the function value which takes the contents of (A) as its argument, and stores it in (C).	5~7	0.75	38.7+2.03n	153.1+11.7n
	166	Dead band	(A) DB (B) → (C)	Finds the value which gives the dead band indicated by (B) for the contents of (A), and stores it in (C).	4~7	0.6	11.7	164.6
	167	Square root	(A)+1·(A) RT (B)	Finds the square root of the double-length data (A)+1·(A), and stores it in (B).	3~6	0.45	88.2	214.1
	168	Integral	(A) INTG (B) → (C)	Calculates the integral for the value of (A) from the integral constant for (B)+1·(B), and stores the result in (C)+1·(C).	4~7	0.6	33.5	288.3
	169	Ramp function	(A) RAMP (B) → (C)	Generates the ramp function for the value of (A) by the parameters starting with (B), and stores it in (C).	4~7	0.6	43.0	256.3
	170	PID	(A) PID (B) → (C)	Carries out the PID calculation for the value of (A) by the parameters starting with (B), and stores it in (C).	4	0.6	890.6	1095.5
	171	Deviation square PID	(A) PID2 (B) → (C)	Carries out the deviation square PID calculation for the value of (A) using the parameters starting with (B), and stores it in (C).	4	0.6	760.7	1035.5
	172	Sine function (SIN)	(A) SIN (B)	Stores in (B) the value obtained by taking the angle (degree) obtained by dividing the value of (A) by 100 and multiplying its sine value by 10000.	3~5	0.45	31.3	152.7
	173	Cosine function(COS)	(A) COS (B)	Stores in (B) the value obtained by taking the angle (degree) obtained by dividing the value of (A) by 100 and multiplying its cosine value by 10000.	3~5	0.45	32.6	154.0

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Function	174	Tangent fungction (TAN)	⊣[(A) TAN (B)]⊢	Stores in (B) the value obtained by taking the angle(degree) obtained by dividing the value of (A) by 100 and multiplying its tangent value by 10000.	3~5	0.45	60.5	168.4
	175	Arc sine function (SIN^{-1})	⊣[(A) ASIN (B)]⊢	Divides the value of (A) by 10000, multiplies the arc sine value by 100, then stores it in (B).	3~5	0.45	34.9	149.1
	176	Arc cosine function (COS^{-1})	⊣[(A) ACOS (B)]⊢	Divides the value of (A) by 10000, multiplies the arc cosine value by 100, then stores it in (B).	3~5	0.45	35.6	149.7
	177	Arc tangent function (TAN^{-1})	⊣[(A) ATAN (B)]⊢	Divides the value of (A) by 10000, multiplies the arc tangent value by 100, then stores it in (B).	3~5	0.45	903.4	1861.1
	178	Exponential function	⊣[(A) EXP (B)+1 (B)]⊢	Finds the exponential of 1/1000 of the absolute value of (A) and stores it in (B)+1•(B)	3~5	0.45	784.4	1212.9
	179	Logarithm	⊣[(A) LOG (B)]⊢	Calculates the common logarithm of the absolute value of (A), multiplies it by 1000 and stores the result in (B).	3~5	0.45	1082.3	1188.6

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Conversion	180	Absolute value	⊣[(A) ABS (B)]⊢	Stores the absolute value of (A) in (B).	3~5	0.45	8.55	114.9
	181	Double-length absolute value	⊣[(A)+1·(A) DABS (B)+1·(B)]⊢	Stores the absolute value of (A)+1·(A) in (B)+1·(B).	3~6	0.45	10.4	151.8
	182	2's complement	⊣[(A) NEG (B)]⊢	Stores the 2's complement of (A) in (B).	3~5	0.45	7.43	113.7
	183	Double-length 2's complement	⊣[(A)+1·(A) DNEG (B)+1·(B)]⊢	Stores the 2's complement of (A)+1·(A) in (B)+1·(B).	3~6	0.45	10.1	151.6
	184	Double-length conversion	⊣[(A) DW (B)+1·(B)]⊢	Converts the signed data in (A) into double-length data, and stores in (B)+1·(B).	3~5	0.45	8.55	130.4
	185	7-segment decode	⊣[(A) 7 SEG (B)]⊢	Converts the 4 bits of (A) into 7-segment code, and stores in (B).	3~5	0.45	7.43	113.7
	186	ASCII conversion	⊣[(A) ASC (B)]⊢	Takes the alphanumerics(maximum 16 aharacters) indicated by (A) and convers them into ASCII code. Stores the result in the location headed by (B).	3~10	0.45	19.8+1.35n	39.4+13.3n
	188	Binary conversion	⊣[(A) BIN (B)]⊢	Converts the BCD data in (A) into binary data and stores it in (B).	3~5	0.45	35.6	141.4
	189	Double-length binary conbersion	⊣[(A)+1·(A) DBIN (B)+1·(B)]⊢	Converts the double-length BCD data in (A)+1·(A) into binary data and stores it in (B)+1·(B).	3~6	0.45	75.2	216.6
	190	BCD conversion	⊣[(A) BCD (B)]⊢	Converts the binaty data in (A) into BCD data and stores it in (B).	3~5	0.45	11.9	118.2
	191	Double-length BCD conbersion	⊣[(A)+1·(A) DBCD (B)+1·(B)]⊢	Converts the binaty data in (A)+1·(A) into BCD data and stores it in (B)+1·(B).	3~6	0.45	42.3	183.8
	204	Floating point conversion	⊣[(A)+1·(A) FLT (B)+1·(B)]⊢	Converts the double-length integer of (A)+1·(A) into floating point data and stores it in (B)+1·(B).	3~5	0.45	25.9	87.9
	205	Fixed point conversion	⊣[(A)+1·(A) FIX (B)+1·(B)]⊢	Converts the floating point data of (A)+1·(A) into double-length integer data and stores it in (B)+1·(B).	3	0.45	80.6	142.5
	206	Floating point absolute value	⊣[(A)+1·(A) FABS (B)+1·(B)]⊢	Stores the absolute value of floating point data of (A)+1·(A) in (B)+1·(B).	3	0.45	11.3	72.1
	207	Floating point sign inversion	⊣[(A)+1·(A) FNEG (B)+1·(B)]⊢	Stores the sign inversion data of floating point data of (A)+1·(A) in (B)+1·(B).	3	0.45	17.6	78.4

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
BCD operation	192	BCD additon	⊣[(A) B+ (B)→(C)]⊢	Carries out BCD addition of the contents of (A) and (B), and stores the result in (C).	4~7	0.6	59.2	212.1
	193	BCD subtraction	⊣[(A) B- (B)→(C)]⊢	Subtracts the contents of (B) from the contents of (A) in BCD, and stores the result in (C).	4~7	0.6	59.2	212.1
	194	BCD multiplication	⊣[(A) B* (B)→(C)+1·(C)]⊢	Multiplies the contents (A) and (B) together in BCD, and stores the result in (C)+1·(C).	4~7	0.6	102.6	271.1
	195	BCD division	⊣[(A) B/ (B)→(C)]⊢	Divides the contents of (A) by the contents of (B) in BCD, and stores the quotient in (C) and the remainder in (C)+1.	4~7	0.6	82.4	250.8
	196	Double-length BCD addition	⊣[(A)+1·(A) DB+ (B)+1·(B)→(C)+1·(C)]⊢	Adds the contents of (B)+1·(B) to the contents of (A)+1·(A) in BCD, and stores the result in (C)+1·(C).	4~9	0.6	112.1	292.7
	197	Double-length BCD subtraction	⊣[(A)+1·(A) DB- (B)+1·(B)→(C)+1·(C)]⊢	Subtracts the contents of (B)+1·(B) from the contents of (A)+1·(A) in BCD, and stores the result in (C)+1·(C).	4~9	0.6	111.6	319.3
	198	Double-length BCD multiplication	⊣[(A)+1·(A) DB* (B)+1·(B)→(C)+1·(C)]⊢	Multiplies the contents of (A)+1·(A) by the contents of (B)+1·(B) in BCD, and stores the result in (C)+3·(C)+2·(C)+1·(C).	4~9	0.6	278.3	482.2
	199	Doble-length BCD division	⊣[(A)+1·(A) DB/ (B)+1·(B)→(C)+1·(C)]⊢	Divides the contents of (A)+1·(A) by the contents of (B)+1·(B) in BCD, and stores the quotient in (C)+1·(C) and the remainder in (C)+3·(C)+2.	4~9	0.6	228.2	432.0
	200	BCD addition with carry	⊣[(A) B+C (B)→(C)]⊢	Adds (B) plus the contents of the carry flag to (A) in BCD, and stores the result in (C). The carry flag changes according to the operation result.	4~7	0.6	60.5	213.5
	201	BCD subtraction with carry	⊣[(A) B-C (B)→(C)]⊢	Subtracts (B) plus the contents of the carry flag from (A) in BCD, and stores the result in (C). The carry flag changes according to the operation result.	4~7	0.6	60.5	213.5
	202	Double-length BCD addition with carry	⊣[(A)+1·(A) DB+C (B)+1·(B)→(C)+1·(C)]⊢	Adds the contents of (B)+1·(B), plus the contents of the carry flag, to(A)+1·(A) in BCD, and stores the result in (C)+1·(C). The carry flag changes according to the operation result.	4~9	0.6	112.7	320.4
	203	Double-length BCD subtraction with carry	⊣[(A)+1·(A) DB-C (B)+1·(B)→(C)+1·(C)]⊢	Subtracts (B)+1·(B) plus the contents of the carry flag from (A)+1·(A) in BCD, and stores the result in (C)+1·(C). The carry flag changes according to the operation result.	4~9	0.6	115.2	322.9

Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Real functions	218	Floating point upper limit	(A)+1·(A) FUL (B)+1·(B)→(C)+1·(C)	Applies the upper limit to the floating point data (A)+1·(A) using (B)+1·(B), and stores the result in (C)+1·(C).	4	0.6	23.6	115.1
	219	Floating point lower limit	(A)+1·(A) FLL (B)+1·(B)→(C)+1·(C)	Applies the lower limit to the floating point data (A)+1·(A) using (B)+1·(B), and stores the result in (C)+1·(C).	4	0.6	23.9	115.4
	220	Floating point dead band	(A)+1·(A) FDB (B)+1·(B)→(C)+1·(C)	Finds the floating point data which gives the dead band by (B)+1·(B) for (A)+1·(A), and stores it in (C)+1·(C).	4	0.6	40.5	132.0
	221	Floating point square root	(A)+1·(A) FRT (B)+1·(B)	Finds the square root of the floating point data (A)+1·(A), and stores it in (B)+1·(B).	3	0.45	288.9	349.8
	222	Floating point PID	(A)+1·(A) FPID (B)+1·(B)→(C)+1·(C)	Carries out the PID calculation for the floating point data (A)+1·(A) using parameters starting with (B)+1·(B), and stores it in (C)+1·(C).	4	0.6	430.0	867.6
	223	Floating point deviation square PID	(A)+1·(A) FPID2 (B)+1·(B)→(C)+1·(C)	Carries out the deviation square PID calculation for the floating point data (A)+1·(A) using parameters starting with (B)+1·(B), and stores it in (C)+1·(C).	4	0.6	478.6	880.6
	224	Floating point sine (SIN)	(A)+1·(A) FSIN (B)+1·(B)	Finds the sine for the floating point data of (A)+1·(A), and stores it in (B)+1·(B).	3	0.45	333.9	711.0
	225	Floating point cosine (COS)	(A)+1·(A) FCOS (B)+1·(B)	Finds the cosine for the floating point data of (A)+1·(A), and stores it in (B)+1·(B)	3	0.45	613.6	753.9
	226	Floating point tangent (TAN)	(A)+1·(A) FTAN (B)+1·(B)	Finds the tangent for the floating point data of (A)+1·(A), and stores it in (B)+1·(B).	3	0.45	755.6	1507.8
	227	Floating point arc sine (SIN^{-1})	(A)+1·(A) FASIN (B)+1·(B)	Finds the arc sine for the floating point data of (A)+1·(A), and stores it in (B)+1·(B).	3	0.45	35.1	559.6
	228	Floating point arc cosine (COS^{-1})	(A)+1·(A) FACOS (B)+1·(B)	Finds the arc cosine for the floating point data of (A)+1·(A), and stores it in (B)+1·(B).	3	0.45	35.1	559.6
	229	Floating point arc tangent (TAN^{-1})	(A)+1·(A) FATAN (B)+1·(B)	Finds the arc tangent for the floating point data of (A)+1·(A), and stores it in (B)+1·(B).	3	0.45	430.0	886.9

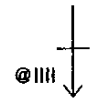
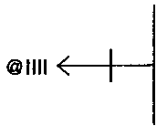
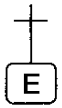
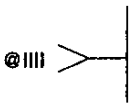
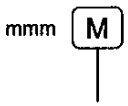
Ladder Diagram Instructions (Function Instructions)

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)		
						Not executed	Minimum	Maximum
Real functions	230	Floating point exponential	⊣[(A)+1·(A) FEXP (B)+1·(B)]⊢	Finds the exponential of the floating point data of (A)+1·(A), and stores it in (B)+1·(B).	3	0.45	678.8	737.3
	231	Floating point logarithm	⊣[(A)+1·(A) FLOG (B)+1·(B)]⊢	Calculates the common logarithm of the floating point data of (A)+1·(A), and stores it in (B)+1·(B).	3	0.45	80.6	1009.9
Input/ output	235	Direct I/O	⊣[I/O (n) (A)]⊢	For the n words registers headed by the input/output register(A), the instruction carries out input/output of data from/to the corresponding I/O module.	3	0.45	20.7+9.7n	—
	236	Expanded data transfer	⊣[(A) XFER (B) → (C)]⊢	Transfers the word block of size (B) from the transfer source indirectly specified by the register(A) to the transfer destination indirectly specified by the register (C).	4	0.6	Dependent on the target	
	237	Special module data read	⊣[(A) READ (B) → (C)]⊢	Carries out data transfer from the memory in the special module to the user area.	4~5	0.6	712+13.5n	—
	238	Special module data write	⊣[(A) WRITE (B) → (C)]⊢	Transfers the contents of the user register area to the memory in the special module.	4~5	0.6	712+13.5n	—

SFC Instructions

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)	
						Inactive	Active
SFC initialize		SFC initialize	Nxx (A) nnnn	When the device (A) has changed from OFF to ON, the instruction inactivates the nnnn steps of the succeeding SFC program, and activates the initial step (SFC activation).	4	5.63	16.43+0.9 ×INT($\frac{n}{16}$-1)
SFC step		Initial step	ssss	Indicates the start of the SFC program and contains action program which correspond on a one-to-one basis. ssss is the step number.	2 (excluding action)	3.83	7.88
		Step	ssss	This is the single unit of control. It contains action program which correspond on a one-to-one basis. ssss is the step number.	1 (excluding action)	0.30	3.00
		End step	ssss	Indicates the end of the SFC program. Returns processing to the corresponding initial step when the immediately preceding transition condition comes true. ssss is the initial step number.	2	0.60	3.15
		Macro step	ssss mmm M	Corresponds on a one-to-one basis to the macro program indicated by mmm. ssss is the step number, and mmm is the macro number.	3	3.37	9.90
		Wait step	ssss (T) xxxx W	Even if the immediately preceding transition condition comes true, this instruction does not carry out the transition until the set period has elapsed. It has action program which correspond on a one-to-one basis. ssss is the step number, (T) is the timer register, and xxxx is the set period.	4 (excluding action)	6.83	9.53
		Alarm step	ssss (T) xxxx (A) A	Monitors the active period, and if the transition has not been made within the set period, sets the alarm device (A) to ON. Contains action program which correspond on a one-to-one basis. ssss is the step number, (T) is the timer register, and xxxx is the set period.	5 (excluding action)	8.18	10.88

SFC Instructions

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (μs)	
						Inactive	Active
Transition		Transition		Indicates the condition for transition between steps. Contains transition condition which correspond on a one-to-one basis.	1 (excluding condition)	0.15	5.62
		SFC End		Indecates the end of SFC program. Jumps to the label indicated by IIII when the transition condition comes true. Contains transition condition which correspond on a one-to-one basis.	2 (excluding condition)	0.30	6.53
		SFC Jump		Indicates jump to desired step. Jumps to the label indicated by IIII when the condition comes true. Contains jump condition which correspond on a one-to-one basis.	5 (excluding condition)	0.75	8.03
		Macro end		Indicates the end of the macro program. Contains transition condition which correspond on a one-to-one basis.	2 (excluding condition)	0.30	6.53
Label		SFC label		Indicates the return destination from the SFC end, or the jump destination from the SFC jump.	2	7.43	11.03
		Macro entry		Indicates start of macro program.	1	0.30	0.30

SFC Instructions

Group	FUN No.	Name	Representation	Summary	Number of steps required	Execution time required (µs)	
						Inactive	Active
Sequence selection		Sequence selection Divergence (I)		From among several connected steps, activates the step for which the transition condition comes true (left priority).	2xn-1 n is the branch count (Excluding transitions, steps, and individual details within the branch)	0.30	6.45
		Sequence selection Divergence (II)				0.30	6.45
		Sequence selection Divergence (III)				0.30	5.77
		Sequence selection Convergence				0.15	0.15
Simultaneous sequences		Simultaneous sequences Divergence (I)		Actuvates all the connected steps.	n+3 n is the branch count (Excluding transitions, steps, and individual details within the branch)	0.15	0.15
		Simultaneous sequences Divergence (II)				0.15	0.15
		Simultaneous sequences Divergence (III)				0.15	0.15
		Simultaneous sequences Convergence (I)				2.20	4.28+0.45n
		Simultaneous sequences Convergence (II)				0.30	2.93+2.93n

A
Annunciator relay 131,170,171,172
Application program (user program) 31,33,141,146
Action program (SFC) 230
Action step (SFC) 229
Automatic I/O allocation 23,89,199
Auxiliary device (R) 158,160
Auxiliary register (RW) 158,160

B
Batch Input/Output processing 83,85
BCD (Binary Coded Decimal) 184
Bit pattern check function 132,169
Block 142,146
Breakpoint function (DEBUG mode) 125

C
Clock-calendar function 168
Comments 141,155
Computer link parameters 144
Contant scan 84
Counter device (C.) 158,161
Counter register (C) 158,161
Cyclic mode (sub-program) 93,98,149,173

D
Data initialization 75,82
Data register (D) 158,161
DEBUG mode 77,124
Debug command 79,80
Device 156
Diagnostics display function 131,170
Digit designation 193
Direct Input / Output processing 86
Direct input device (I) 158,160
Direct input register (IW) 158,160
Direct output device (O) 158,160
Direct output register (OW) 158,160
Double length BCD 187
Double length integer 186

E
Edged modification 219
EEPROM 71,102,139
End step (SFC) 226
ERROR mode 77,104
Event history 108
Execution time measurement function 116
Expanded file register 103,139

F
File register (F) 158,162
Floating point data 188
Floating scan 83,95
Force function 123
Force RUN command 18,78
Function block 216
Function instruction 219
Functional specifications 72

H
HALT (operation mode switch) 18,78
HALT command 18,78
HALT mode 77
HOLD mode 77
Hot restart function 113

I
i designation 85,201
I/O allocation 23,198
I/O allocation information 32,145,198
I/O allocation rule 26,203
I/O disable function (DEBUG mode) 125
I/O interrupt 100,150
I/O module with interrupt function 150
I/O mounting check 82
IC memory card 71,103,139
Index modification 189
Index register (I,J,K) 158,162,189
Initial load 20,75,82,102,103
Initial step (SFC) 225
Input device (X) 158,159,198
Input register (XW) 158,159,198

Integer 183
Interrupt assignment information 145,152
Interrupt enable/disable 101
Interrupt program 100,150

L
Ladder diagram 211,214
Link device (Z) 158,162
Link register (LW) 158,162
Link register (W) 158,162
Link Relay (L) 158,162

M
Macro end (SFC) 228
Macro entry (SFC) 228
Macro program (SFC) 223
Macro step (SFC) 227
Main program 92,147
Manual I/O allocation 23,201
Memory capacity 143
Memory protect function 110
Mode control 76,83
Mode transition condition 78,79,80
Module type 24,199
Multitask function 92,236

N
N scans execution function (DEBUG mode) 129
Network assignment information 206

O
One time mode (sub-program) 93,96,149,173
Online I/O replacement function 115
Online program changing function 90,123
Operation modes 17,76,77
Operation mode switch 18,76,78
Output device (Y) 158,159,198
Output register (YW) 158,159,198

P
P-RUN (operation mode switch) 18,78,110
Password function 135

Peripheral support 88
PLC control commands 18,78,79
Power interruption decision 74
Power interruption shutdown function 111
Program execution sequence 217,218
Program ID 143
Program read from EEPROM/IC memory card 91,102,103
Program size setting 143,155
Program type 33,92,141
Program write into EEPROM/IC memory card 91,102,103
Programming language 211
Programming precautions for Ladder diagram 219,220
Programming precautions for multitask 236,237
Programming precautions for SFC 233,234,235

R
RAM/ROM switch 18,78,79
RAS function 104
Real time clock function (clock-calender function) 168
Register 156
Register data validity check function 133,169
Register / device address range 36,37,158
Register / device initialization 39,75,82
Retentive memory area 39,144
RUN (operation mode switch) 18,78
RUN command 18,78
RUN mode 77
RUN-F mode 77
Rung 34,215
Rung number 33,214

S
Sampling buffer 117
Sampling trace function 117
Scan control 81
Scan cycle 21,106,116
Scan mode 83
Scan time setting 84,144
Self diagnostic check 104
Sequence selection (SFC) 226
Sequential function chart (SFC) 212,221
SFC 212,221

SFC capacity limit....233
SFC end....225
SFC initialization....224
SFC jump....228
SFC label....229
SFC main program....222
Simultaneous sequences (SFC)....227
Single rung execution function (DEBUG mode)....128
Single step execution function (DEBUG mode)....126
Special device (S)....158,161,163
Special register (SW)....158,161,163
Status latch function....122
Stop (SFC)....225
Stop number (SFC)....222
Stops used....143
Stop condition setting function (DEBUG mode)....129
Sub-program....92,93,148
Sub-program execution time....95,144
Sub-routine....153
System comments....143
System configuration....71
System information....32,143
System initialization....74

T
Timer device (T.)....87,158,161
Timer interrupt....100,144,150
Timer interrupt interval....144
Timer register (T)....87,158,161
Timer update....87
Timing relay....88,167
Transition (SFC)....225

U
Unit base address setting function....202
Unsigned double-length integer....185
Unsigned integer....183
User data....36,156
User data initialization....39,75,82
User program....31,141
User program check....82
User program execution....21,83

User program memory.....31,71,141

W
Wait step (SFC) 229
Watchdog timer check 107

TOSHIBA

TOSHIBA INTERNATIONAL (EUROPE) LTD.

1 Roundwood Avenue
Stockley Park, Uxbridge
Middlesex UB11 1AR, ENGLAND
Tel:081 848 4466 Fax:081 848 4969
Tlx:265062 TSB LDN G Cable:Toshibatic London.

TOSHIBA INTERNATIONAL CORPORATION

indystrial Equipment Division
13131 West Little York Road
Houston. Texas 77041, U.S.A.
Tel:713-466-0277 1-800-231-1412 Telex:762078

TOSHIBA INTERNATIONAL CORPORATION PTY. LTD.

Unit 1,9 Orion Road, Lane Cove
N.S.W.2066, AUSTRALIA
Tel:02-428-2077 Telex:AA25192

TOSHIBA CORPORATION

Industrial Equipment Department
1-1, Shibaura 1-chome, Minato-ku, Tokyo 105, Japan
Tel:(03)3457-4900
Cable:TOSHIBA TOKYO